ACADEMIC PRESS

Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Information and Computation 186 (2003) 104–139

Information and Computation

www.elsevier.com/locate/ic

Some connections between bounded query classes and non-uniform complexity☆

Amihood Amir,[a,1] Richard Beigel,[b,2] and William Gasarch[c,*,3]

[a] *Department of Math and Comp. Sci., Bar-Ilan University, Ramat Gan 52900, Israel*
[b] *Department of CIS, Temple University, Wachman Hall, 1805 N Broad Street, Philadelphia, PA 19122, USA*
[c] *Department of C.S. and Inst. for Adv. Comp. Stud., University of Maryland, Room 3213, A. V. Williams Building, College Park, MD 20742, USA*

Received 25 July 1994; revised 20 March 2003

Abstract

Let $A(x)$ be the characteristic function of A. Consider the function $C_k^A(x_1, \ldots, x_k) = A(x_1) \cdots A(x_k)$. We show that if C_k^A can be computed in polynomial time with fewer than k queries to some set X then $A \in \mathrm{P/poly}$. A generalization of this result has applications to bounded query classes, circuits, and enumerability. In particular we obtain the following. (1) Assuming $\Sigma_3^p \neq \Pi_3^p$, there are functions computable using $f(n)+1$ queries to SAT that are not computable using $f(n)$ queries to SAT, for $f(n) = \mathrm{O}(\log n)$. (2) If C_k^A, restricted to length n inputs, can be computed by an unbounded fanin oracle circuit of size $s(n)$ and depth $d(n)$, with $k-1$ queries to some set X, then A can be computed with an unbounded fanin (non-oracle) circuit of size $n^{\mathrm{O}(k)}s(n)$ and depth $d(n)+\mathrm{O}(1)$. (3) Assuming that $\mathrm{PH} \neq \Sigma_4^p \cap \Pi_4^p$, and $\epsilon < 1$, #SAT is not 2^{n^ϵ}-enumerable.

☆ This paper is dedicated to the memory of Ronald V. Book, 1937–1997.

* Corresponding author. Fax: +1-301-405-6707.

E-mail addresses: gasarch@cs.umd.edu (W. Gasarch), amir@cs.biu.ac.il (A. Amir), beigel@cis.edu (R. Beigel).

[1] Research performed at University of Maryland and Georgia Tech. Supported in part by NSF Grants CCR-8803641 and CCR-96101709.

[2] Research performed at Johns Hopkins, Yale University, The University of Maryland, Lehigh University, and DIMACS. Supported in part by the National Science Foundation under Grants CCR–8958528, CCR–9415410, and CCR-9877150; in part by DIMACS, an NSF Science and Technology Center funded under Contract STC-91-19999 and by the NJ Commission on Science and Technology; and in part by the Human–Computer Interaction Laboratory under NASA Grant NAG 52895.

[3] Supported in part by NSF Grants CCR-8803641, CCR-9020079, CCR-9301339, and CCR-9732692.

0890-5401/$ - see front matter © 2003 Elsevier Science (USA). All rights reserved.
doi:10.1016/S0890-5401(03)00091-9

1. Introduction

The standard complexity classes (e.g., P, NP) were defined to classify sets. To classify functions several researchers [8,28,45] have defined complexity classes based on the *number of queries* that a polynomial-time algorithm has to make to a particular set. For example, Krentel [45] has shown that to compute the chromatic number of a graph, $\Theta(\log n)$ queries to SAT are necessary and sufficient (assuming P $\neq$ NP). Krentel [45] and Gasarch [28] have classified many functions in terms of queries to SAT. (See [32] for a survey of such results.) We review the definitions of these bounded query classes in Section 2.2.

There have been many papers on bounded query classes. Several papers (including this one) have shown that it is unlikely for various bounded query hierachies to collapse [1,10,11,14,17,20,24,26,38,48, 50,60,66] Bounded queries have been related to questions ostensibly unrelated to counting oracle queries [25].

The notion of bounded queries has also been studied in computability theory. See [8,29] for the first work in the field, [12,13,15,46] for the most important papers in the field, [30,31], for two surveys, and [33] for a book.

A natural function to look at in this context is the following:

Definition 1.1. If $A \subseteq \{0,1\}^*$ and $k \in \mathsf{N}$ then $C_k^A : (\{0,1\}^*)^k \to \{0,1\}^k$ is defined by

$$C_k^A(x_1, \ldots, x_k) = A(x_1) \cdots A(x_k),$$

where $A(x)$ is the characteristic function of A: 1 if $x \in A$, 0 otherwise.

The function C_k^A is easily seen to be computable with k parallel queries to A. When can it be computed with $k-1$ queries to A? to some other set X? Subsequent to this work, this question has been studied in [1,14,50,60].

In Sections 3 and 4 we show that if C_k^A can be computed with $k-1$ queries to some X then A is easy in some sense. In particular, we show the following.

- If $(\exists k)(\exists X)$ such that $C_{2^k}^A$ can be computed with k queries to X then $A \in$ P/poly. In addition, $A \in \mathrm{EL}_2$, the second level of the extended low hierarchy (defined in [6] and in Definition 3.1 of this paper). (Corollary 3.9).
- If $(\exists k)(\exists X)$ such that C_k^A can be computed with $k-1$ queries to X then $A \in$ P/poly. (Theorem 4.2).

In Sections 5, 6 and 7 we use these theorems, and the techniques used to obtain them, to extend (though not generalize) previously known theorems about bounded query classes, circuits, and enumerability. We state the previously known theorems and our extensions (for readability we do not state the strongest form of either the previous results or of ours):

- Krentel [45] showed that if P $\neq$ NP then, for any increasing $f \in$ PF, $f(n) \leqslant (1-\epsilon)\log n$, $f(n)$ queries to SAT are more powerful than $f(n)-1$. [4] We show that if $\Sigma_3^p \neq \Pi_3^p$ then for any $f(n) = \mathrm{O}(\log n)$, $f(n)$ queries to SAT are more powerful than $f(n)-1$ (Corollary 5.2).
- Cai [21] showed that if a constant depth oracle circuit computes k length-n instances of PARITY, with only $k-1$ queries, it must have size $2^{n^{\Omega(1)}}$. We show that if there are a circuits of size $s(n)$ and depth $d(n)$ that compute $C_{k(n)}^{A=n}$ while making only $k(n)-1$ queries to some set X then there are (non-oracle)

[4] Krentel actually showed this just for $f(n) \leqslant \frac{1}{2}\log n$, but his proof can be modified to $f(n) \leqslant (1-\epsilon)\log n$. See [10].

circuits of size $n^{O(k(n))}s(n)$ and depth $d(n) + O(1)$ that compute $A^{=n}$. By applying the lower bound on parity [19,35–37,71] we easily obtain Cai's result (Corollary 6.4).

- Cai and Hemachandra [23] proved that if $P \neq P^{\#P}$ then, for all k, #SAT is not n^k-enumerable, i.e., there is no polynomial algorithm that, with formula ψ as input, produces n^k numbers one of which is #SAT(ψ). We extend their definition of enumerability to superpolynomial functions and prove the following

$$(\forall \epsilon, 0 < \epsilon < 1)[\#SAT \ \ 2^{n^\epsilon}\text{-enumerable} \Rightarrow \#SAT \in PF^{\Sigma_4^p \cap \Pi_4^p}]$$

(Corollary 7.11). In addition we obtain the above-mentioned result of Cai and Hemachandra by an entirely different method (Corollary 7.12).[5]

There is a longer version of this paper [3]. When there is a result that is in that paper but not in this one, the reader will be informed. The following types of results are in the long version but not in this version.

(i) Generalizations and extensions of results in this paper. Usually the generalization uses the same ideas as in this paper but requires more detail.

(ii) If A is weakly p-selective then $A \in P/poly$ and $A \in EL_2$. (A slightly stronger theorem was already known [44].) If A is NPMV-selective then $A \in NP/poly$. If A is NPSV-selective then $A \in NP/poly \cap$ co-NP/poly $\cap EL_3$. (Subsequent to this work it was shown that p-selective sets are in NP *linear* [39]. For more on p-selective sets see [40].)

(iii) Examination of the *classes* of sets A that have properties based on how easy it is to compute C_k^A. In particular we study p-selective sets, closure properties, p-genericity, and the structure of the degrees of such sets.

2. Definitions and useful facts

Section 2.1 reviews some standard definitions from complexity theory. Section 2.2 reviews some definitions and facts relevant to bounded query classes.

2.1. Definitions from the literature

Notation 2.1. Throughout this paper Σ is a fixed alphabet and $\% \notin \Sigma$. We use $\langle x_1, \ldots, x_n \rangle$ to mean $x_1 \% x_2 \cdots x_{n-1} \% x_n$.

Definition 2.2. PF is the class of functions that can be computed in polynomial time.

(i) If $A \subseteq \Sigma^*$ then PF^A (P^A) is the class of functions (sets) that can be computed in polynomial time using oracle A. The number of steps it takes to ask "$y \in A$" is $|y|$.

(ii) If $f : \Sigma^* \to \Sigma^*$ then PF^f (P^f) is the class of functions (sets) that can be computed in polynomial time using oracle f. The number of steps it takes to ask "what is $f(y)$" is $|y| + |f(y)|$.

(iii) If $\mathcal{C}$ is a class of sets or a class of functions then $PF^{\mathcal{C}}$ ($P^{\mathcal{C}}$) is the class of functions (sets) that can be computed in polynomial time using an oracle from $\mathcal{C}$.

[5] Cai and Hemachandra obtained the result independent of ourselves, at roughly the same time.

Definition 2.3. A set A is *polynomial truth table reducible* to B, denoted $A \leqslant_{\mathrm{tt}}^{p} B$, if there exists $f \in \mathrm{PF}$ such that (i) for all x, $f(x) = \langle y_1, \ldots, y_m, \varphi \rangle$ where $y_i \in \Sigma^*$ and φ is an m-place Boolean formula, and (ii) $x \in A$ iff $\varphi(b_1, \ldots, b_m)$ is true, where b_i is the truth value of "$y_i \in B$." The number m is the *norm* of $f(x)$. Two sets A and B are *polynomial truth table equivalent* if $A \leqslant_{\mathrm{tt}}^{p} B$ and $B \leqslant_{\mathrm{tt}}^{p} A$. This is denoted by $A \equiv_{\mathrm{tt}} B$. Note that m and $|\phi|$ are bounded by a polynomial in $|x|$.

Definition 2.4. Let k be a constant. $A \leqslant_{k-\mathrm{tt}}^{p} B$ if $A \leqslant_{\mathrm{tt}}^{p} B$ via a function f such that, for all x, $f(x)$ has norm $\leqslant k$. $A \leqslant_{\mathrm{btt}}^{p} B$ if there exists a constant k such that $A \leqslant_{k-\mathrm{tt}}^{p} B$.

Definition 2.5. If $\mathcal{C}$ is a class of sets then A is $\leqslant_{\mathrm{tt}}^{p}$-hard for $\mathcal{C}$ if for every $B \in \mathcal{C}$, $B \leqslant_{\mathrm{tt}}^{p} A$. The notions of $\leqslant_{\mathrm{btt}}^{p}$-hard and $\leqslant_{k-\mathrm{tt}}^{p}$-hard are defined similarly.

Notation 2.6. If A and B are sets then $A \oplus B$ is the set $\{1x : x \in A\} \cup \{0x : x \in B\}$.

Pippenger [53] showed that the class of languages recognized by polynomial-size circuits is P/poly (though it was not quite defined yet). This inspired Karp and Lipton [42] to define general advice classes.

Definition 2.7. A function h has *polynomial-size output* if there exists a polynomial p such that $(\forall x)$ $[|h(x)| \leqslant p(|x|)]$. Note that there are no constraints on how difficult it is to compute h.

Definition 2.8. Let $\mathcal{C}$ be a class of functions. A function f is in $\mathcal{C}$/poly if there exists $g \in \mathcal{C}$ and a function h with polynomial size output such that $(\forall n)(\forall x)[|x| = n \Rightarrow f(x) = g(x, h(0^n))]$. The function h is called the *advice function* and $h(0^n)$ is called the *advice for strings of length n*. We use the phrase w *serves as advice for* f *on strings of length* $\leqslant n$ if $h(0^n) = w$.

Cai and Hemachandra [22] defined $b(n)$-enumerability as follows.

Definition 2.9. Let $b(n)$ be a function with range N. Let $\% \notin \Sigma$. A function $f : \Sigma^* \to \Sigma^*$ is $b(n)$-*enumerable* if there exists $e \in \mathrm{PF}$, $e : \Sigma^* \to (\Sigma \cup \{\%\})^*$, such that, for all x, $e(x)$ is a list of at most $b(|x|)$ elements of Σ^*, separated by %, at least one of which is $f(x)$.

This definition only makes sense if $b(n)$ is bounded by a polynomial. We define a more general notion of enumerability that allows superpolynomial b.

Definition 2.10. Let $b(n)$ be a function with range N. A function f is $b(n)$-*enumerable* if there exists $e \in \mathrm{PF}$, $e : \Sigma^* \times \mathsf{N} \to \Sigma^*$, such that, for all x, there exists an $i < b(|x|)$ such that $e(x, i) = f(x)$. (We need to have $i < b(|x|)$ instead of $i \leqslant b(|x|)$ since the natural numbers N include 0.) We assume the second input to e is written in binary.

Definition 2.11. The quantifier $\overset{\infty}{\forall} x$ means "for all but a finite number of x." The domain of x will usually be Σ^*.

Definition 2.12. Let $i \geqslant 1$. QBF_i is the set of true quantified Boolean formulas that have an $\exists$ as the leftmost quantifier and make at most $i - 1$ alternations of quantifiers. Note that $\mathrm{QBF}_1 = \mathrm{SAT}$. It is well

known that QBF_i is complete for Σ_i^p [64,70]. Let $\mathrm{QBF} = \bigcup_{i=1}^{\infty} \mathrm{QBF}_i$. It is well known that QBF is complete for PSPACE [65].

Definition 2.13. A *$\mathcal{G}$-circuit on n variables* is a directed acyclic graph with n input nodes of in-degree 1 and one output node of out-degree 1. The nodes that are neither inputs or outputs are called *gates* and are labelled with Boolean functions from the set $\mathcal{G}$. A $\mathcal{G}$-circuit computes a Boolean function in the usual way. Its *size* is the number of gates in it. Its *depth* is the length of the longest path from an input to the output. Throughout this paper we assume that the gate set $\mathcal{G}$ includes a NOT-gate as well as AND-gates and OR-gates of every (i.e., unbounded) fanin. Sometimes it will not matter what other elements $\mathcal{G}$ contains; then by convention we will abuse notation and call a $\mathcal{G}$-circuit simply a *circuit*.

Definition 2.14. A *$\mathcal{G}$-circuit family* is a collection $\{D_n\}_{n=1}^{\infty}$ of $\mathcal{G}$-circuits where D_n is a $\mathcal{G}$-circuit on n inputs. A $\mathcal{G}$-circuit family where each D_n has size $\leqslant s(n)$ and depth $\leqslant d(n)$ is called an $(s(n), d(n))$ *$\mathcal{G}$-circuit family*. We will continue the tradition of abusing notation by calling a $\mathcal{G}$-circuit family just a $\mathcal{G}$-circuit.

2.2. Bounded query classes

Bounded query classes were defined in [4,11] as follows.

Definition 2.15.

- If A is an oracle and $j(n)$ is a function from N to N then $\mathrm{PF}^A_{j(n)-\mathrm{T}}$ is the class of functions that can be computed by a polynomial time oracle Turing machine that makes, on inputs of length n, at most $j(n)$ queries to oracle A. (We call such queries *serial*. Book and Ko [18] call them adaptive.)
- If A is an oracle and $j(n)$ is a function from N to N then $\mathrm{PF}^A_{j(n)-\mathrm{tt}}$ is the class of functions that can be computed by a polynomial time oracle Turing machine that, on inputs of length n, prepares a list of the $j(n)$ queries it is going to make to A *before* actually making any of them. (We call such queries *parallel*. Book and Ko [18] call them non-adaptive.)
- $\mathrm{P}^A_{j(n)-\mathrm{T}}$ is the class of all B such that $\chi_B \in \mathrm{PF}^A_{j(n)-\mathrm{T}}$.
- $\mathrm{P}^A_{j(n)-\mathrm{tt}}$ is the class of all B such that $\chi_B \in \mathrm{PF}^A_{j(n)-\mathrm{tt}}$.

Note. The oracle A in the definition above will usually be a set, but the definitions also hold when the oracle is a function.

If f is computable by making $j(n)$ oracle queries, then there are only $2^{j(n)}$ *possible* values for the result of f. The informal notion of *possibility* is made precise by using the notion of enumerability as defined in Section 2.1.

The connection between bounded queries and enumerability is formalized by the following two facts from [9, Lemma 3.2].

Fact 2.16. *Let f be any function. Let $j(n) \in \mathrm{PF}$. The following are equivalent*:

(i) *There exists X such that $f \in \mathrm{PF}^X_{j(n)-\mathrm{T}}$.*

(ii) *f is $2^{j(n)}$-enumerable*.

(iii) *There exists* Y *such that* $f \in \mathrm{PF}^{Y}_{j(n)-\mathrm{tt}}$.(*If* $2^{j(n)}$ *is bounded by a polynomial then* $Y \in \mathrm{P}^{f}_{1-\mathrm{tt}}$.)

By plugging in C^A_k for f and $j(n) = j$ (a constant) into Fact 2.16 we obtain the following.

Fact 2.17. *Let* $j, k \in \mathsf{N}$. *The following are equivalent*:
(i) *There exists* X *such that* $C^A_k \in \mathrm{PF}^{X}_{j-\mathrm{T}}$.
(ii) C^A_k *is* 2^j*-enumerable*.
(iii) *There exists* $B \in \mathrm{P}^{A}_{k-\mathrm{tt}}$ *such that* $C^A_k \in \mathrm{PF}^{B}_{j-\mathrm{tt}}$.

We are interested in finding out when the function C^A_k *requires* k queries to A (or any oracle X). On the other hand, we are also interested in determining when the function C^A_k can be computed with far fewer than k queries. The following definitions reflect these two extreme notions.

Definition 2.18.
(i) A set A is *p-terse* if, for all k, $C^A_k \notin \mathrm{PF}^{A}_{(k-1)-\mathrm{T}}$.
(ii) A set A is *p-superterse* if, for all sets X, for all k, $C^A_k \notin \mathrm{PF}^{X}_{(k-1)-\mathrm{T}}$.
(iii) Let k be a constant. A set A is *k-cheatable* if there exists a set X such that $C^A_{2^k} \in \mathrm{PF}^{X}_{k-\mathrm{T}}$.
(iv) A is *cheatable* if A is k-cheatable for some constant k. Note that by Fact 2.17 a set is k-cheatable iff $C^A_{2^k}$ is 2^k-enumerable.

We state some known useful consequences of a set being k-cheatable. We need a known combinatorial fact that we will use both here and later.
N"

Fact 2.19 ([11, 16, 51])**.** *Let* $\mathcal{C}$ *be a collection of* m *sets. There exists a set* X *such that*
$\bullet\ (\forall S, S' \in \mathcal{C})[S \neq S' \Rightarrow S \cap X \neq S' \cap X]$.
$\bullet\ |X| \leqslant m - 1$.

Fact 2.20. *Let* $k \in \mathsf{N}$ *and* $A \subseteq \Sigma^*$.
(i) *If* A *is* k*-cheatable then* $(\forall m)[C^A_m \in \mathrm{PF}^{A}_{(2^k-1)-\mathrm{tt}}$ *via a machine that, on input* $\{x_1, \ldots, x_m\}$, *queries a subset of* $\{x_1, \ldots, x_m\}$]. *This reduction is uniform in* m. (*Theorem* 5.4.*i of* [*11*].)
(ii) *If* A *is* k*-cheatable then* $(\exists X)(\forall m)[C^A_m \in \mathrm{PF}^{X}_{k-\mathrm{T}}]$.(*Theorem* 5.4.*ii of* [*11*].)
(iii) A *is* k*-cheatable iff* $(\forall m)[C^A_m$ *is* 2^k*-enumerable*].

The following fact about p-superterse sets will point the way to a generalization of p-terseness that is used in Theorem 4.4.

Fact 2.21. *If* A *is not p-superterse then there exists a* $k \in \mathsf{N}$ *and* $w \in \mathrm{PF}$, $w : (\Sigma^*)^k \to \{0,1\}^k$, *such that for all* $x_1, \ldots, x_k$, $C^A_k(x_1, \ldots, x_k) \neq w(x_1, \ldots, x_k)$.

Proof. Let k be such that there exists X, $C^A_k \in \mathrm{PF}^{X}_{(k-1)-\mathrm{T}}$. By Fact 2.17 C^A_k is 2^{k-1}-enumerable. We compute w as follows: on input $(x_1, \ldots, x_k)$ we compute all 2^{k-1} possibilities for $C^k_A(x_1, \ldots, x_k)$ and output the least element (using lexicographical ordering) of $\{0,1\}^k$ that is not one of them. □

The converse of Fact 2.21 is also known, i.e., if such a w exists then A is not p-superterse (see [9]).

3. Cheatable sets

If a set is cheatable then it should be easy in some sense. In this chapter we pin down that intuition. In Section 3.1 we prove a powerful lemma (Lemma 3.7) about computations with bounded queries. From it we obtain that if A is cheatable then $A \in \mathrm{P/poly}$ and $A \in \mathrm{EL}_2$ (see Definition 3.1). Sets in P/poly are easy in that they reduce to sparse sets. Sets in EL_2 are easy since, if A is in EL_2, then $\Sigma_2^{p,A} \subseteq \mathrm{NP}^{A\oplus\mathrm{SAT}}$, so A does not add much to the strength of Σ_2^p.

In Section 3.2 we show that if A is cheatable and self-reducible then $A \in \mathrm{P}$. As a corollary we obtain that NP-hard sets are not cheatable (unless $\mathrm{P} = \mathrm{NP}$).

3.1. Circuits and lowness

Schöning[56] defined the low and high hierarchies to classify NP sets. Balcázar, Book, and Schöning [6] defined the extended low and extended high hierarchies to classify sets in general. (All these notions are analogous to similar concepts in computability theory. See [47].)

Definition 3.1. Let $k \geqslant 1$ and A be a set. The set A is in EL_k, the kth level of the extended low hierarchy, if $\Sigma_k^{p,A} \subseteq \Sigma_{k-1}^{p,A\oplus\mathrm{SAT}}$. The set A is in EH_k, the kth level of the extended high hierarchy, if $\Sigma_k^{p,A\oplus\mathrm{SAT}} \subseteq \Sigma_k^{p,A}$.

The following facts from [56,57] will aid the intuition that sets in EL_k are easy and sets in $\mathrm{EH_k}$ are hard.

Fact 3.2.
(i) $\mathrm{P} \subseteq \mathrm{EL}_1 \subseteq \mathrm{EL}_2 \cdots$.
(ii) $\cdots \mathrm{EH}_3 \subseteq \mathrm{EH}_2 \subseteq \mathrm{EH}_1$.
(iii) *If* $(\exists k)[\mathrm{SAT} \in \mathrm{EL}_k]$ *then* PH *collapses*.
(iv) *If* $(\exists k)[\emptyset \in \mathrm{EH_k}]$ *then* PH *collapses*.
(v) $\mathrm{NP} \cap \text{co-NP} \subseteq \mathrm{EL}_1$.
(vi) $\mathrm{P/poly} \subseteq \mathrm{EL}_3$.

The classification of sets into these classes gives a sense of how hard those sets are. In this paper we will show that cheatable sets are in P/poly, and hence in EL_3. We will then show that more can be said: cheatable sets are actually in EL_2.

We need the following lemma. The techniques to prove it are standard but do not seem to be in the literature.

Notation 3.3. Let $R(x, y)$ be a relation. The expression $B = \{x : (\exists^p y)[R(x, y)]\}$ means that there exists a polynomial q such that

$$B = \{x : (\exists y)[|y| \leqslant q(|x|) \wedge R(x, y)]\}.$$

The expression $C = \{x : (\forall^p y)[R(x, y)]\}$ means that there exists a polynomial q such that

$$C = \{x : (\forall y)[|y| \leqslant q(|x|) \Rightarrow R(x, y)]\}.$$

This notation can be extended to more variables.

It is well known that if $B \in \Sigma_i^{p,A}$ then there exists a relation $R^A \in \mathrm{P}^A$ such that

$$B = \{x : (\exists^p y_1)(\forall^p y_2) \cdots (Q^p y_i)[R^A(x, y_1, \ldots, y_i)]\}.$$

(Q^p is $\exists^p$ if i is odd, and is $\forall^p$ if i is even.)

Lemma 3.4. *Let $A \in \mathrm{P/poly}$. Let $p(n)$ be the length of the advice. We assume that p is $1-1$. Let C be a set of strings that satisfy the following properties*:
(i) *If $w \in C$ then $(\exists n)[|w| = p(n)]$ and w could serve as advice for the* P/poly *algorithm for A^n.*
(ii) *For all n there exists $w \in C$ such that $|w| = p(n)$.*
If $C \in \mathrm{P}^{A\oplus\mathrm{SAT}}$ then $A \in \mathrm{EL}_2$.

Proof. We show $\Sigma_2^{p,A} \subseteq \Sigma_1^{p,A\oplus\mathrm{SAT}}$. Let $B \in \Sigma_2^{p,A}$. There exists a relation $R^A \in \mathrm{P}^A$ such that $B = \{x : (\exists^p y)(\forall^p z)[R^A(x, y, z)]\}$.

Let $R'(w, x, y, z)$ denote the result of trying to compute $R^A(x, y, z)$ by assuming that w is advice for A, hence answering all queries to A by using the P/poly algorithm for A and advice w.

Note that

$$B = \{x : (\exists^p w)(\exists^p y)[w \in C \wedge (\forall^p z)[R'(w, x, y, z)]]\}.$$

Since $C \in \mathrm{P}^{A\oplus\mathrm{SAT}}$ and $R' \in \mathrm{P}$, $B \in \Sigma_1^{p,A\oplus\mathrm{SAT}}$. □

Definition 3.5. Let X and Y be two disjoint sets. If Z is such that $X \subseteq Z$ and $Y \subseteq \overline{Z}$ then Z *separates X from Y.*

Notation 3.6. If A is a set and $k \in \mathbb{N}$ then A^k is $A \times \cdots \times A$ where the number of A's is k. (The "$\times$" denotes cartesian product.)

Lemma 3.7. *Let A be a set. Assume there exist k and Z such that the following hold.*
(i) $Z \subseteq (\Sigma^*)^k$
(ii) *Z separates A^k from $\{(x_1, \ldots, x_k) : (\forall i \neq j)[x_i \neq x_j] \wedge |A \cap \{x_1, \ldots, x_k\}| = k-1\}$.*
(iii) $Z \in \mathrm{P}^A_{(k-1)-\mathrm{T}}$ *via an algorithm $\mathcal{A}$ that queries only components of the input.*
Then $A \in \mathrm{P/poly}$. If, in addition, $\overline{A}$ satisfies the same condition then $A \in \mathrm{EL}_2$.

The proof of membership in P/poly is largely inspired by Ko [44].

Proof. Algorithm $\mathcal{A}$ takes as input a k-tuple $(x_1, \ldots, x_k)$ and makes $\leqslant k-1$ queries to A, which are all in $\{x_1, \ldots, x_k\}$. Without loss of generality, we assume that (1) if $x_1, \ldots, x_k$ are input and are distinct then algorithm $\mathcal{A}$ makes exactly $k-1$ distinct queries to A, and (2) the queries made are independent of the order of the inputs (e.g., if on input (x_1, x_2, x_3) the queries are x_1 and x_2 then on input (x_3, x_2, x_1)

the queries are x_1 and x_2). If $\mathcal{A}$ does not have these properties then we can modify it as follows: on input $(x_1, \ldots, x_k)$ first sort them lexicographically and then run $\mathcal{A}$ on this sorted list. Let $g(\{x_1, \ldots, x_k\})$ be the set of oracle queries asked by $\mathcal{A}$ on input $(x_1, \ldots, x_k)$. Note that $g(\{x_1, \ldots, x_k\})$ is a $(k-1)$-element subset of $\{x_1, \ldots, x_k\}$.

For any set $S \subseteq A \cap \Sigma^n$ we will show how to find a $(k-1)$-element set $X \subseteq S$ such that knowing X allows us to verify that a constant fraction of the elements of S are in A. By iterating this procedure $O(\log |S|)$ times we will generate polynomial advice that allows us to verify that each element of S is in A. We will generate such advice for $S = A \cap \Sigma^n$; then we can test strings in Σ^n for membership in A because all the strings that cannot be verified as being in A will be in $\overline{A}$.

If $x_1, \ldots, x_{k-1} \in A$ then, for all x,

$$\mathcal{A}(x_1, \ldots, x_{k-1}, x) = 1 \Rightarrow x \in A,$$
$$\mathcal{A}(x_1, \ldots, x_{k-1}, x) = 0 \Rightarrow x \notin A.$$

If $x_1, \ldots, x_{k-1}$ are fixed and the value of $C^A_{k-1}(x_1, \ldots, x_{k-1})$ is known, then perhaps we can use $\mathcal{A}$ in an algorithm for A. Unfortunately the queries made by $\mathcal{A}(x_1, \ldots, x_{k-1}, x)$ might include x itself. We seek $x_1, \ldots, x_{k-1}$ such that, for many x, $\mathcal{A}(x_1, \ldots, x_{k-1}, x)$ does not query x.

In general let $[S]^k$ denote the set of all k-element subsets of S, and let s denote $|S|$. For $S \subseteq A$ and $X = \{x_1, \ldots, x_{k-1}\} \in [S]^{k-1}$, we define the set of strings for which $(x_1, \ldots, x_{k-1})$ is useful advice:

$$\text{ADVISEES}(X) = \{x \in S - X : g(X \cup \{x\}) = X\}.$$

Note that if we know $(x_1, \ldots, x_{k-1})$ then we can use algorithm $\mathcal{A}$ to verify $x \in A$ for every element $x \in \text{ADVISEES}(X)$ since $\mathcal{A}(x_1, \ldots, x_{k-1}, x)$ does not query x.

In the following calculation we use the fact that every $Y \in [S]^k$ can be partitioned into $X \in [S]^{k-1}$ and $x \in \text{ADVISEES}(X)$ via $g(Y) = X$ and $x \in Y - X$.

$$\begin{aligned}\sum_{X \in [S]^{k-1}} |\text{ADVISEES}(X)| &= \sum_{X \in [S]^{k-1}} \sum_{x \in \text{ADVISEES}(X)} 1 \\ &= \sum_{X \cup \{x\} \in [S]^k} 1 \\ &= |[S]^k|.\end{aligned}$$

Therefore, there exists $X \in [S]^{k-1}$ such that

$$|\text{ADVISEES}(X)| \geqslant \frac{|[S]^k|}{|[S]^{k-1}|} = \frac{\binom{s}{k}}{\binom{s}{k-1}} = \frac{s-k+1}{k}.$$

For this choice of $X = \{x_1, \ldots, x_{k-1}\}$, the tuple $(x_1, \ldots, x_{k-1})$ is useful advice for $(s-k+1)/k$ strings in addition to the members of X; hence it is useful advice for $(s-k+1)/k + k - 1 = (s + k^2 - 2k + 1)/k \geqslant s/k$ strings.

In particular, let $S_0 = A \cap \Sigma^n$. As shown in the preceding paragraph, there is a $(k-1)$-tuple X_1 that is useful advice for at least $|S_0|/k$ strings in S_0. Let $S_1 = S_0 - (\text{ADVISEES}(X_1) \cup X_1)$. Then $|S_1| \leqslant |S_0|/c$, where $c = k/(k-1)$. Repeat the argument using S_1 in place of S_0 to obtain X_2 and S_2. Repeat for p

iterations, stopping when $|S_p| < k$. Then $p \leqslant \log_c(|\Sigma|^n) = O(n)$. Let d be such that $p \leqslant dn$. Note that d is independent of n and of what is happening at this stage. Note that

$$X_1 \cup \cdots \cup X_p \cup S_p \subseteq A \cup \Sigma^n \subseteq X_1 \cup \cdots \cup X_p \cup S_p \cup \text{ADVISEES}(X_1) \cup \cdots \cup \text{ADVISEES}(X_p).$$

Our advice for strings of length n consists of the sets $X_1, \ldots, X_p$ and S_p. This advice contains at most $(p+1)(k-1)(n)$ bits, which is $O(n^2)$ because k is a constant and $p = O(n)$.

If y is a string of length n, then we determine if $y \in A$ as follows.

Step 1: If $y \in X_1 \cup \cdots \cup X_p \cup S_p$ then output(YES) and halt.
Step 2: For $i = 1$ to p
(a) Let z_i be the k-tuple containing the elements of $X_i \cup \{y\}$ in lexicographic order.
(b) We simulate $\mathcal{A}$ on input z_i using the answer "yes" for each query. If $\mathcal{A}$ queries only elements of X_i and outputs b, then output(b) and halt. (Note that if $\mathcal{A}$ queries only elements of X_i then the queries are answered correctly above, so the simulation correctly distinguishes between whether all k components of z_i or only $k-1$ of them are in A. Since X_i is a $(k-1)$-element subset of A, this tells us whether $y \in A$.) If $\mathcal{A}$ queries y then go to the next value of i. (If $x \in A$ then, by construction, one of the i will work.)
Step 3: Output(NO). (By the construction, every element of A will be recognized in the previous step. Hence if no such i exists then $y \notin A$.)

We now assume that $\overline{A}$ satisfies the condition of the theorem as well. Let the analogous algorithm be $\overline{\mathcal{A}}$. We will conclude that $A \in \text{EL}_2$. First, build advice $Y_1, \ldots, Y_{p'}, T_{p'} \subseteq \overline{A}$ similar to the procedure above. We now use the sets $X_1, \ldots, X_p, S_p, Y_1, \ldots, Y_{p'}, T_{p'}$ to show $A \in \text{P/poly}$ via an algorithm that satisfies the property of Lemma 3.4; therefore, $A \in \text{EL}_2$.

If y is a string of length n, then we determine if $y \in A$ as follows.

Step 1: If $y \in X_1 \cup \cdots \cup X_p \cup S_p$ then output(YES) and halt. If $y \in Y_1 \cup \cdots \cup Y_p \cup T_p$ then output(NO) and halt.
Step 2: For $i = 1$ to p
(a) Let z_i be the k-tuple containing the elements of $X_i \cup \{y\}$ in lexicographic order.
(b) We simulate $\mathcal{A}$ on input z_i using the answer "yes" for each query. If $\mathcal{A}$ queries only elements of X_i and outputs b, then output(b) and halt. (Note that if $\mathcal{A}$ queries only elements of X_i then the queries are answered correctly above, so the simulation correctly distinguishes between whether all k components of z_i or only $k-1$ of them are in A. Since X_i is a $(k-1)$-element subset of A, this tells us whether $y \in A$.) If $\mathcal{A}$ queries y then go to the next value of i. (By construction, if $y \in A$, then one of the i will work.)
Step 3: For $i = 1$ to p
(a) Let z_i be the k-tuple containing the elements of $Y_i \cup \{y\}$ in lexicographic order.
(b) We simulate $\overline{\mathcal{A}}$ on input z_i using the answer "yes" for each query. (Note that these are queries to $\overline{A}$.) If $\overline{\mathcal{A}}$ queries only elements of Y_i and outputs b, then output(b) and halt. (Note that if $\overline{\mathcal{A}}$ queries only elements of Y_i then the queries are answered correctly above, so the simulation correctly distinguishes between whether all k components of z_i or only $k-1$ of them are in $\overline{A}$. Since Y_i is a $(k-1)$-element subset of $\overline{A}$, this tells us whether $y \in \overline{A}$.) If $\overline{\mathcal{A}}$ queries y then go to the next value of i. (By construction, if $y \in \overline{A}$, one of the i will work.)

The algorithm is more complicated than is needed to recognize A, but the point is that the set of strings that can serve as advice will be relatively simple. Let ADV_n be the set of all strings that can

serve as advice for strings of length n using this algorithm. We give a $\mathrm{P}^{A\oplus\mathrm{SAT}}$ algorithm to decide a subset of $\bigcup_{n=0}^{\infty} ADV_n$ that has, for each n, at least one string of length n. By Lemma 3.4 this shows $A \in \mathrm{EL}_2$.

A string that claims to be advice is of the form $X_1, \ldots, X_p, S_p, Y_1, Y_2, \ldots, Y_{p'}, T_{p'}$ where $S_p \cup \bigcup_{i=1}^{p} X_i \subseteq A$ and $T_{p'} \cup \bigcup_{i=1}^{p'} Y_i \subseteq \overline{A}$. These inclusions can easily be verified with queries to A. It suffices to verify that for every $y \in \Sigma^n$ either (1) y belongs to the set $\bigcup_{i=1}^{p} X_i \cup \bigcup_{i=1}^{p} Y_i \cup S_p \cup T_p$, (2) there exists an i such that $g(X_i \cup \{y\}) = X_i$ (we can test this since $X_i \subseteq A$), or (3) there exists an i such that $g(Y_i \cup \{y\}) = Y_i$ (we can test this since $Y_i \subseteq \overline{A}$). This is a co-NP predicate and hence can be answered with a query to SAT. □

Theorem 3.8. *If there exists $k \in \mathsf{N}$ such that $C_k^A \in \mathrm{PF}_{(k-1)-\mathrm{tt}}^A$ via an algorithm $\mathcal{A}$ that queries only components of the input, then $A \in \mathrm{P/poly}$ and $A \in \mathrm{EL}_2$.*

Proof. Lemma 3.7 is satisfied with the values A, $Z = A^k$, and k. Hence $A \in \mathrm{P/poly}$.

Lemma 3.7 is also satisfied with the values $\overline{A}$, $Z = \overline{A}^k$, and k. Hence $A \in \mathrm{EL}_2$. □

Corollary 3.9. *If A is cheatable then $A \in \mathrm{P/poly}$ and $A \in \mathrm{EL}_2$.*

Proof. By Fact 2.20 if A is k-cheatable then $C_{2^k}^A \in \mathrm{PF}_{(2^k-1)-\mathrm{tt}}^A$ via an algorithm that queries only components of the input. Therefore Theorem 3.8 applies. □

In the proof of Lemma 3.7 we only used the fact that Z was computable in *polynomial time* in Step 2b of the first algorithm. If Z is in a class $\mathcal{C}$ then perhaps we can obtain $A \in \mathcal{C}/\mathrm{poly}$. This train of thought leads to a powerful theorem from which we can prove some known theorems about p-selective sets.

Definition 3.10. A language Z is in $\mathrm{NP}_{k-\mathrm{T}}^A$ if there exists a non-deterministic polynomial time oracle Turing machine $M^{()}$ such that M^A recognizes Z and on *each* computation path makes at most k queries.

Lemma 3.11. *Let $A \subseteq \Sigma^*$. Assume there exist $k \in \mathsf{N}$ and $Z \subseteq (\Sigma^*)^k$ such that Z separates A^k from $\{(x_1, \ldots, x_k) : (\forall i \neq j)[x_i \neq x_j] \wedge |A \cap \{x_1, \ldots, x_k\}| = k-1\}$. If $Z \in \mathcal{C}_{(k-1)-T}^A$ where $\mathcal{C}$ is* NP *or any deterministic time or space class that contains* P *and has $\mathcal{C} = \mathrm{P}_{\mathrm{O}(n)-\mathrm{T}}^{\mathcal{C}}$ (e.g.,* PSPACE *or* EXPTIME*) then $A \in \mathcal{C}/\mathrm{poly}$.*

Proof. First we obtain advice exactly as in the proof of Lemma 3.7. Let the advice be $X_1, \ldots, X_p, S_p$.

Case 1. $\mathcal{C}$ is a deterministic complexity class that contains P. Proceed exactly as in the proof of Lemma 3.7.

Case 2. $\mathcal{C} = \mathrm{NP}$. Then $Z \in \mathrm{NP}_{k-T}^A$ via non-deterministic algorithm $\mathcal{A}$.

Let $\mathcal{A}(y, X_i, d)$ be the result of simulating $\mathcal{A}$ on non-deterministic path d with input the tuple consisting of the elements of $X_i \bigcup \{y\}$ in lexicographic order, with the following provisos.

- If y is queried then we reject.
- All queries (to strings in X_i) are answered yes.

From the nature of the advice we have

$$y \in A \text{ iff } \left(y \in S_p \text{ or } (\exists i)(\exists d)[y \in X_i \vee \mathcal{A}(y, X_i, d) = \text{YES}]\right).$$

Hence $A \in \text{NP/poly}$. □

Note. A more general version of Lemma 3.11 is in the long version [3].

3.2. Self-reduction and NP-hardness

In this section we show that if a set is self-reducible and cheatable then it is in P (This result was first stated (without proof) in [8], crediting the current authors. We subsequently learned that [34] obtained the result independently.) We use this to show that, under a suitable hypothesis, certain sets A are not cheatable. We then prove a lemma that extends this to any set B such that $A \leqslant_{\text{T}}^{p} B$.

Intuitively, a set A is self-reducible if the question "$x \in A$?" can be reduced to questions of the form "$y \in A$?" where $|y| < |x|$. Many natural sets, *e.g.*, most NP-complete sets in [27], are self-reducible. Schnorr [55] was the first to define the concept. We use an alternative definition which is more general and is implicit in the literature. It was first introduced by [49].

Definition 3.12. Let p be a function and $\prec$ be an ordering on Σ^*. The ordering $\prec$ has *p-bounded chains* if whenever $x_m \prec x_{m-1} \prec \cdots \prec x_1$, we have $m \leqslant p(|x_1|)$ and $(\forall i)[|x_i| \leqslant p(|x_1|)]$. The ordering $\prec$ has *polynomially-bounded chains* if there exists a polynomial p such that $\prec$ has p-bounded chains.

Definition 3.13. A set A is *polynomial Turing self-reducible* (henceforth self-reducible) if there exists a polynomial-time computable partial order $\prec$ such that

(i) $\prec$ has polynomially bounded chains; and
(ii) there exists a polynomial-time bounded oracle Turing machine $M^{()}$ such that
  (a) all strings queried by $M^{()}$ precede the input string in the ordering $\prec$, and
  (b) the language accepted by M^A is A.

The definition of *polynomial truth-table self-reducible* (henceforth tt-self-reducible) is similar, just replace the polynomial Turing reduction with a polynomial tt-reduction.

We state a theorem that appeared in [11]. Theorems 3.18 and 3.25 are generalizations of it.

Proposition 3.14. *A is tt-self-reducible and cheatable iff A is in P.*

In order to improve Proposition 3.14 from tt-self-reducible to self-reducible we need the following lemma.

Definition 3.15. A *tree* is a finite subset of $\{0, 1\}^*$ that is closed under prefix. A Σ^*-*labeled tree* is a tree where every node is mapped to an element of Σ^*.

Definition 3.16. Let $f \leqslant_{\text{T}}^{p} B$ via an oracle Turing machine $M^{()}$ that runs in time bounded by $p(n)$ for all oracles. Let $x \in \Sigma^*$. We view an answer of YES as 1, and an answer of NO as 0. The *oracle query tree for $M^{()}(x)$* is the labeled tree such that the node $b_1 b_2 \cdots b_{m-1}$ ($b_i \in \{0, 1\}$) is mapped to the m^{th}

question that would be asked if the first $m-1$ questions are answered $b_1, b_2, \ldots, b_{m-1}$. Note that the depth of the tree is at most $p(|x|)$.

Lemma 3.17. *If B is k-cheatable and $f \leqslant_{\mathrm{T}}^{p} B$ then $f \leqslant_{(2^k-1)-\mathrm{tt}}^{p} B$ using queries that belong to the oracle query tree in the $f \leqslant_{\mathrm{T}}^{p} B$ Turing reduction.*

Proof. Let $f \leqslant_{\mathrm{T}}^{p} B$ via an oracle Turing machine $M^{()}$ that runs in time bounded by $p(n)$ for all oracles. Since B is k-cheatable we have, by Fact 2.20(iii), that $C_{2^{k+2}}^{B}$ is 2^k-enumerable. We use this later.

Given x, we show how to compute $f(x)$ with 2^k-1 parallel queries to B where those queries are on the oracle query tree. Assume $|x| = n$. We generate the oracle query tree for $M^{()}(x)$ and prune it so that it remains small. Let T_i be the pruned tree through level i (note that T_0 is the one-node tree that labels that one node with the first query). Let $a(i)$ be the number of nodes in T_i and $b(i)$ be the number of leaves in T_i. We describe the pruning process and derive the bounds $a(i) \leqslant 2^{k+1}$ and $b(i) \leqslant 2^k$. Note that $a(0) = b(0) = 1$ trivially.

Inductively assume that we have constructed T_i with $a(i) \leqslant 2^{k+1}$ and $b(i) \leqslant 2^k$. Find the queries asked on both a YES and NO answer to the leaf queries in T_i. The total number of queries is now $\leqslant a(i) + 2b(i) \leqslant 2^{k+2}$. Assume, without loss of generality, that there are exactly 2^{k+2} queries (we can repeat queries to pad). Let the queries be $x_1, \ldots, x_{2^{k+2}}$. Since $C_{2^{k+2}}^{B}$ is 2^k-enumerable we can generate 2^k possibilities for $C_{2^{k+2}}^{B}(x_1, \ldots, x_{2^{k+2}})$. Each possibility generated can be mapped to the leaf that those answers would lead to (and also to an answer to the leaf node query, which we ignore). Prune the leaves that do not correspond to any possibility. The number of leaves left is $b(i+1) \leqslant 2^k$. This bound on the number of leaves implies that the total number of nodes is $a(i+1) \leqslant 2^{k+1}$.

The tree $T_{p(n)}$ can be found in polynomial time and has at most 2^{k+1} queries. Let $x_1, \ldots, x_{2^{k+1}}$ be those queries. Note that all of the x_i are on the oracle query tree of the $f \leqslant_{\mathrm{T}}^{p} B$ Turing reduction. By Fact 2.20.i $C_{2^{k+1}}^{B}(x_1, \ldots, x_{2^{k+1}})$ can be determined by querying 2^k-1 of the elements of $\{x_1, \ldots, x_{2^{k+1}}\}$. Once this is done $f(x)$ can easily be computed. So $f \leqslant_{(2^k-1)-\mathrm{tt}}^{p} B$ and all the queries came from the oracle query tree of the $f \leqslant_{\mathrm{T}}^{p} B$ Turing reduction. □

Theorem 3.18. *A is self-reducible and cheatable iff A is in P.*

Proof. Assume A is self-reducible and cheatable. By Proposition 3.14 we need only prove that A is tt-self-reducible. Let $M^{()}$ be such that A is self-reducible via M^A. Note that for all x, for all queries y made in the oracle query tree of $M^{()}(x)$, $y \prec x$. Let k be such that A is k-cheatable. Apply Lemma 3.17 to the set A and the characteristic function χ_A to obtain that $\chi_A \in \mathrm{PF}_{(2^k-1)-\mathrm{tt}}^{A}$ using queries that belong to the oracle query tree of $M^{()}(x)$. Since all such queries are $\prec x$, A is tt-self-reducible. The converse is trivial. □

Corollary 3.19.

(i) *If* P $\neq$ NP *then* SAT *is not cheatable.*
(ii) *If* P $\neq \Sigma_i^p$ *then* QBF_i *is not cheatable.*
(iii) *If* P $\neq$ PSPACE *then* QBF *is not cheatable.*

(*See Definition* 2.12 *for a definition of* QBF_i *and* QBF.)

Proof. It is well known that SAT (QBF_i, QBF) is self reducible and complete for NP (Σ_i^p, PSPACE). Hence the corollary follows from Theorem 3.18. □

Corollary 3.19 (i) can be restated as follows: if $\mathrm{P} \neq \mathrm{NP}$ and A is NP-hard under polynomial m-reductions then A is not cheatable. (The other parts can be restated in similar ways.) We want to extend this to sets A that are NP-hard under polynomial T-reductions. For this we need the following lemma.

Lemma 3.20. *If B is k-cheatable and $A \leqslant_{\mathrm{T}}^p B$ then A is k-cheatable.*

Proof. Since $A \leqslant_{\mathrm{T}}^p B$, $C_{2^k}^A \leqslant_{\mathrm{T}}^p B$. By Lemma 3.17 $C_{2^k}^A \in \mathrm{PF}_{(2^k-1)-\mathrm{tt}}^B$. Since B is k-cheatable, the $2^k - 1 \leqslant 2^k$ parallel queries can be answered by using k queries to some set X. Hence $C_{2^k}^A$ can be computed with k queries to that set X, so A is k-cheatable. □

Corollary 3.21.
(i) *If* $\mathrm{P} \neq \mathrm{NP}$ *then all* $\leqslant_{\mathrm{T}}^p$*-hard sets for* NP *are not cheatable.*
(ii) *If* $\mathrm{P} \neq \Sigma_i^p$ *then all* $\leqslant_{\mathrm{T}}^p$*-hard sets for* Σ_i^p *are not cheatable.*
(iii) *If* $\mathrm{P} \neq \mathrm{PSPACE}$ *then all* $\leqslant_{\mathrm{T}}^p$*-hard sets for* PSPACE *are not cheatable.*
(iv) *All* $\leqslant_{\mathrm{T}}^p$*-hard sets for* EXPTIME *are not cheatable.*

Proof. Parts (i)–(iii) follow from Corollary 3.19 and Lemma 3.20. We prove iv. Assume, by way of contradiction, that A is a cheatable set that is $\leqslant_{\mathrm{T}}^p$-hard for EXPTIME. By an easy diagonalization there exists a p-superterse $B \in \mathrm{EXPTIME}$. Clearly $B \leqslant_{\mathrm{T}}^p A$. By Lemma 3.20 B is cheatable. But no set can be both cheatable and superterse. □

Corollary 3.19 (i) (though not Corollary 3.21) has been superseded by [14,50] who have shown that if $\mathrm{P} \neq \mathrm{NP}$ then any set that is btt-hard for NP is p-superterse. In a different direction [60] has shown the following. Assume that there is a procedure that will, given $c \log n$ formulas $(\phi_1, \ldots, \phi_{c \log n})$ where $(\forall i)[|\phi_i| \leqslant n]$, eliminate one possiblity for $C_{c \log n}^{\mathrm{SAT}}(\phi_1, \ldots, \phi_{c \log n})$. Then the promise problem Unique-SAT is in P, and hence by [10], $\mathrm{NP} = \mathrm{R}$.

3.3. Non-constant number of queries

The results of the last section can be extended to a variation on k-cheatability where k is a function instead of a constant. The full proof of this is in the long version [3]; however, we give the definitions and results.

Definition 3.22. Let A be a set and let q be a function. The function C_q^A is defined only on the domain $\bigcup_{m=0}^{\infty} (\Sigma^{\leqslant m})^{q(m)}$ as follows

$$(\forall m)(\forall x_1, \ldots, x_{q(m)} \in \Sigma^{\leqslant m})[C_q^A(x_1, \ldots, x_{q(m)}) = A(x_1) \cdots A(x_{q(m)})].$$

Notation 3.23. When we write $C_q^A(x_1, \ldots, x_{q(m)})$ we assume $(\forall i)[|x_i| \leqslant m]$.

In this section we avoid using the bounded query classes notation and the enumerability notation since the parameter of interest will be m, which is *not* the length of the input.

Definition 3.24. Let A be a set. Let $k(m) = \mathrm{O}(\log m)$. Let $q(m) = 2^{k(m)}$. A is $k(m)$-*cheatable* if there exists a set X and a polynomial time oracle Turing machine $M^{()}$ such that, for all m, M^X computes $C_q^A(x_1, \ldots, x_{q(m)})$ with at most $k(m)$ queries to X.

Theorem 3.25. *A is self-reducible and* $\mathrm{O}(\log m)$*-cheatable iff A is in* P.

Corollary 3.26.
(i) *If* $\mathrm{P} \neq \mathrm{NP}$ *then* SAT *is not* $\mathrm{O}(\log m)$*-cheatable*.
(ii) *If* $\mathrm{P} \neq \Sigma_i^p$ *then* QBF_i *is not* $\mathrm{O}(\log m)$*-cheatable*.
(iii) *If* $\mathrm{P} \neq \mathrm{PSPACE}$ *then* QBF *is not* $\mathrm{O}(\log m)$*-cheatable*.
(*See Definition* 2.12 *for a definition of* QBF_i *and* QBF.)

Corollary 3.26 (i) can be restated as follows: if $\mathrm{P} \neq \mathrm{NP}$ and A is NP-hard under polynomial m-reductions then A is not $k(m)$-cheatable. (The other parts can be restated in similar ways.) We want to extend this to sets A that are NP-hard under polynomial T-reductions.

Lemma 3.27. *If B is* $\mathrm{O}(\log m)$*-cheatable and $A \leqslant_{\mathrm{T}}^p B$ then A is* $\mathrm{O}(\log m)$*-cheatable*.

Corollary 3.28.
(i) *If* $\mathrm{P} \neq \mathrm{NP}$ *then* $\leqslant_{\mathrm{T}}^p$*-hard sets for* NP *are not* $\mathrm{O}(\log m)$*-cheatable*.
(ii) *If* $\mathrm{P} \neq \Sigma_i^p$ *then* $\leqslant_{\mathrm{T}}^p$*-hard sets for* Σ_i^p *are not* $\mathrm{O}(\log m)$*-cheatable*.
(iii) *If* $\mathrm{P} \neq \mathrm{PSPACE}$ *then* $\leqslant_{\mathrm{T}}^p$*-hard sets for* PSPACE *are not* $\mathrm{O}(\log m)$*-cheatable*.
(iv) *All* $\leqslant_{\mathrm{T}}^p$*-hard sets for* EXPTIME *are not* $\mathrm{O}(\log m)$*-cheatable*.

4. Non-p-superterse sets

We pursue the question "Which sets A can be non-p-superterse?" In Section 4.1 we show that if A is not p-superterse then $A \in \mathrm{P/poly}$. We then extend the techniques to obtain a generalization that involves a non-constant number of queries. This generalization is applied in Sections 5, 6, 7 to obtain results about bounded query classes, circuits, and enumeration.

4.1. Circuits

We prove that if A is not p-superterse then A is in P/poly. Although the result resembles Lemma 3.7 the proof is substantially different.

Notation 4.1.
- If t is a k-tuple of bits $(b_1, \ldots, b_k)$ then $t[i]$ denotes b_i and $t[i : j]$ denotes $(b_i, \ldots, b_j)$.
- If x is a string, y is a j-tuple of strings $(y_1, \ldots, y_j)$, and z is a k-tuple of strings $(z_1, \ldots, z_k)$ then (y, z) denotes $(y_1, \ldots, y_j, z_1, \ldots, z_k)$, (x, y) denotes $(x, y_1, \ldots, y_j)$ and (y, x) denotes $(y_1, \ldots, y_j, x)$.

Theorem 4.2. *If A is not p-superterse then $A \in \mathrm{P/poly}$.*

Proof. Let A be a non-p-superterse set, that is, $(\exists X)(\exists k)[C_k^A \in \mathrm{PF}^X_{(k-1)-\mathrm{T}}]$. By Fact 2.21 there exists $w \in \mathrm{PF}$, $w : (\Sigma^*)^k \to \{0,1\}^k$, such that for all tuples t in $(\Sigma^*)^k$, $w(t) \neq C_k^A(t)$. Hence we have the weaker statement, denoted by $S(k)$, that there exists $w \in \mathrm{PF/poly}$, $w : \bigcup_{n=1}^{\infty}(\Sigma^n)^k \to \{0,1\}^k$, such that for all $t \in (\Sigma^n)^k$, $w(t) \neq C_k^A(t)$. We show that for all $m \geqslant 2$, $S(m)$ implies $S(m-1)$. Since we have $S(k)$, this implies $S(1)$, which yields $A \in \mathrm{P/poly}$. Since we only need the implication $S(m) \Rightarrow S(m-1)$ for $m \leqslant k$ we can treat m as a constant.

Assume $S(m)$ is true via w. We show $S(m-1)$ by constructing $w' \in \mathrm{P/poly}$, $w' : \bigcup_{n=1}^{\infty}(\Sigma^n)^{m-1} \to \{0,1\}^{m-1}$, such that $w'(t) \neq C_{m-1}^A(t)$. We show how to construct the advice for computing $w'(t)$ where $t \in (\Sigma^n)^{m-1}$. The construction is an iterative process that, at each iteration, finds advice for computing w' for a fraction of its inputs (this uses the function w). Two things may happen in the construction. If the construction finds advice for every tuple in $(\Sigma^n)^{m-1}$ then we have advice for w' on $(\Sigma^n)^{m-1}$. If the construction is unable to find advice then the very reason that it cannot find advice yields a probabilistic algorithm for $A \cap \Sigma^n$. More advice removes the probability. (We will use Theorem A.6 from the appendix, which is a variation of Schöning's proof that $\mathbf{BP} \cdot \mathcal{C} \subseteq \mathcal{C}/\mathrm{poly}$.)

Convention. If $z \in \Sigma^n$, $t = (t_1, \ldots, t_{m-1}) \in (\Sigma^n)^{m-1}$ then $w(z, t_1, \ldots, t_{m-1})$ will be denoted by $w(z,t)$.

Definition. Let $n, m \in \mathsf{N}$. Assume $S(m)$ is true via w. Let $z \in \Sigma^n$. $\mathrm{ADVISEES}(z)$ is the set of all $(m-1)$-tuples $t \in (\Sigma^n)^{m-1}$ such that $w(z,t)[1] = A(z)$. (Since we know that $w(z,t) \neq C_m^A(z,t)$ we have $w(z,t)[2:m] \neq C_{m-1}^A(t)$.) We say that a string z is *advice for a set T* of $(m-1)$-tuples if

$$|\mathrm{ADVISEES}(z) \cap T| > \frac{1}{4}|T|.$$

CONSTRUCTION OF ADVICE for Σ^n

$T_n := (\Sigma^n)^{m-1}$
$Z_n := \emptyset$
While there exists a string z in Σ^n that is advice for T_n
    choose such a z
    $T_n := T_n - \mathrm{ADVISEES}(z)$
    $Z_n := Z_n \cup \{z\}$

END OF CONSTRUCTION

Note that after the ith iteration $|T_n| \leqslant (\frac{3}{4})^i * |(\Sigma^n)^{m-1}|$. Hence there are at most $\mathrm{O}(\log |(\Sigma^n)^{m-1}|) = \mathrm{O}(n)$ iterations. Since the number of elements in Z_n is bounded by the number of iterations, $|Z_n| = \mathrm{O}(n)$.

Let I be the set of all n such that $T_n \neq \emptyset$. We show the following.

(i) If $n \notin I$ then there exists $w_1' \in \mathrm{P/poly}$, $w_1' : \bigcup_{n \notin I}(\Sigma^n)^{m-1} \to \{0,1\}^{m-1}$, such that $w_1'(t) \neq C_{m-1}^A(t)$.

(ii) If $n \in I$ then there exists $w_2' \in \mathrm{P/poly}$, $w_2' : \bigcup_{n \in I}(\Sigma^n)^{m-1} \to \{0,1\}^{m-1}$, such that $w_2'(t) \neq C_{m-1}^A(t)$.

These two easily yield the desired w'.

(i) If $n \notin I$ then $T_n = \emptyset$. Let the advice be the union of $\{(z, A(z)) : z \in Z_n\}$ and the advice needed to compute w on $(\Sigma^n)^m$ (we use the induction hypothesis here). For $t \in \bigcup_{n \notin I}(\Sigma^n)^{m-1}$ we define $w_1'(t)$ by

$$z = \min\{y : y \in Z_n \text{ and } t \in \text{ADVISEES}(y)\},$$
$$w_1'(t) = w(z,t)[2:m].$$

(The minimum in the definition of z is with respect to lexicographic ordering.) Thus,

$$(\forall n \notin I)(\forall t \in (\Sigma^n)^{m-1})[w_1'(t) \neq C_{m-1}^A(t)].$$

w_1' can be computed efficiently using the advice.

(ii) If $n \in I$ then $T_n \neq \emptyset$ and for all $z \in \Sigma^n$ at least 3/4 of the elements $t \in T_n$ satisfy $w(z,t)[1] \neq A(z)$. Let $B_n = \{\langle x,t\rangle : w(x,t)[1] = 0\}$. Note that for all $n \in I$, for all $x \in \Sigma^n$

$$x \in A \Rightarrow \Pr[\langle x,t\rangle \in B_n : t \in T_n] \geqslant \frac{3}{4},$$
$$x \notin A \Rightarrow \Pr[\langle x,t\rangle \notin B_n : t \in T_n] \geqslant \frac{3}{4}.$$

We would like to say that we have a probabilistic algorithm for A and hence, by known techniques, $A \in \text{P/poly}$. There are two objections to this: (1) we may have each case applying infinitely often, so we have a probablistic argument infinitely often, and a direct argument infinitely often and (2) we have our string t ranging over T_n, not over Σ^n. The first objection is not serious: the case that we are in can be part of the advice. The second objection requires a modification of the proof that BPP $\subseteq$ P/poly. Such a modification appears in the appendix.

Since $w \in \text{P/poly}$ we have $\bigcup_{n\in I} B_n \in \text{P/poly}$. By Theorem A.6 (with $Y = \bigcup_{n\in I} T_n$ and $B = \bigcup_{n\in I} B_n$) $A \cap \bigcup_{n\in I} \Sigma^n \in (\text{P/poly})/\text{poly} = \text{P/poly}$. Hence the desired w_2' can easily be constructed. □

Because all sets in P/poly belong to EL_3, it follows that all non-p-superterse sets belong to EL_3. Hoene and Nickelsen [41, Theorem 6, Corollary 7] have shown that this is optimal.

Corollary 4.3.

(i) *If* $\Sigma_2^p \neq \Pi_2^p$ *then all* $\leqslant_{\text{T}}^p$*-hard sets for* NP *are p-superterse.*
(ii) *If* $\Sigma_2^p \neq$ PSPACE *then all* $\leqslant_{\text{T}}^p$*-hard sets for* PSPACE *are p-superterse.*
(iii) *If* P $\neq$ PSPACE *then all* $\leqslant_{\text{tt}}^p$*-hard sets for* PSPACE *are p-superterse.*
(iv) *Every* $\leqslant_{\text{tt}}^p$*-hard set for* EXPTIME *is p-superterse.*

Proof.

(i) If NP $\subseteq$ P/poly then $\Sigma_2^p = \Pi_2^p$ [42,43].
(ii) If PSPACE $\subseteq$ P/poly then $\Sigma_2^p =$ PSPACE [42].
(iii) Assume that A is $\leqslant_{\text{tt}}^p$-hard for PSPACE and A is not p-superterse. Then $\Sigma_2^p =$ PSPACE by (ii). Hence A is also $\leqslant_{\text{tt}}^p$-hard for Δ_2^p. In [9] Beigel showed that if a non-p-superterse set is $\leqslant_{\text{tt}}^p$-hard for Δ_2^p then P = NP. By [64] this implies P $= \Sigma_2^p$, hence we have P $= \Sigma_2^p =$ PSPACE.
(iv) Assume that A is $\leqslant_{\text{tt}}^p$-hard for EXPTIME and that A is not p-superterse. Since every set in P/poly is tt-reducible to a tally set [18][Theorem 3.6], there exists a tally set T with $A \leqslant_{\text{tt}}^p T$. So T is $\leqslant_{\text{tt}}^p$-hard for EXPTIME; however, no tally set can be $\leqslant_{\text{tt}}^p$-hard for EXPTIME [18, Theorem 6.1]. □

Corollary 4.3 (i) has been superseded by [14,50], who have shown that if P $\neq$ NP then SAT is p-superterse.

4.2. Non-constant number of queries

We extend Theorem 4.2 to a non-constant number of queries. In the proof of Theorem 4.2 we used Fact 2.21: if A is not p-superterse then there is $k \in \mathsf{N}$ and $w \in \mathrm{PF}$, $w : (\Sigma^*)^k \to \{0,1\}^k$, such that for all $x_1, \ldots, x_k$, $C_k^A(x_1, \ldots, x_k) \neq w(x_1, \ldots, x_k)$. Theorem 4.4 can be seen as an extension of Theorem 4.2 by replacing the function w with a relation W. (We originally proved Theorem 4.4 for a function w. Pankaj Rohatgi pointed out to us that the same proof works for relations.)

We will use the results of this section in Sections 5, 6, and 7.

We omit the proof of the next theorem since it is similar to the proof of Theorem 4.2 and is in the long version [3].

Theorem 4.4. *Let $k(n)$ be polynomial-bounded and let A be a language. Assume there exists a set $W \subseteq \bigcup_{n=0}^{\infty}[(\Sigma^n)^{k(n)} \times \{0,1\}^{k(n)}]$ such that for every $k(n)$-tuple t of length-n strings*

- *there exists $\vec{b} \in \{0,1\}^{k(n)}$ such that $(t, \vec{b}) \in W$, and*
- *for every $\vec{b} \in \{0,1\}^{k(n)}$, if $(t, \vec{b}) \in W$ then $C_{k(n)}^A(t) \neq \vec{b}$.*

Then $A \in \mathrm{NP}^W/\mathrm{poly} \cap \text{co-NP}^W/\mathrm{poly}$.

Theorem 4.4 holds for any set W. If we restrict W and examine the proof, we can obtain

Lemma 4.5. *If A is self-reducible and $A \in \mathrm{NP}^W/\mathrm{poly}$ then $\Sigma_2^{p,A} \subseteq \Sigma_3^{p,W}$, hence $A \in \Sigma_3^{p,W} \cap \Pi_3^{p,W}$ and $\mathrm{P}^A \subseteq \Sigma_3^{p,W} \cap \Pi_3^{p,W}$.*

Corollary 4.6. *Let $A, k(n)$ and W be as in Theorem* 4.4.

(a) *If $W \in \mathrm{DTIME}(T(n))$ then $A \in \mathrm{DTIME}(n^{\mathrm{O}(k(n))}T(n))/\mathrm{poly}$.*

(b) *If W is accepted by an $(s(n), d(n))$ $\mathcal{G}$-circuit family (the nth $\mathcal{G}$-circuit operates on inputs $t \in (\Sigma^n)^{k(n)} \times \{0,1\}^{k(n)}$) then A is accepted by an $(n^{\mathrm{O}(k(n))}s(n), d(n) + \mathrm{O}(1))$ $\mathcal{G}$-circuit family (the nth $\mathcal{G}$-circuit operates on Σ^n).*

(c) *If A is self-reducible then $A \in \Sigma_3^{p,W} \cap \Pi_3^{p,W}$ and $\mathrm{P}^A \subseteq \Sigma_3^{p,W} \cap \Pi_3^{p,W}$.*

5. Applications: bounded query classes

We apply Theorem 4.4 to obtain results about bounded query classes. In particular we show that under complexity-theoretic assumptions (e.g., $\Sigma_3^p \neq \Pi_3^p$) there are separations between bounded query classes. We actually prove stronger results from which separations will be clear. For example, we show that if $\Sigma_3^p \neq \Pi_3^p$ and $f(n) = \mathrm{O}(\log n)$ then, for all X, $\mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}$; hence, under these hypotheses, $\mathrm{PF}^{\mathrm{NP}}_{(f(n)-1)-\mathrm{tt}} \subset \mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{tt}}$ and $\mathrm{PF}^{\mathrm{NP}}_{(f(n)-1)-\mathrm{T}} \subset \mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{T}}$.

Theorem 5.1. *Let A be a set. Let $f, k \in \mathrm{PF}$ be such that the following hold.*

(i) $(\overset{\infty}{\forall} m)[f(mk(m)) \leqslant k(m)]$.

(ii) $(\exists X)[\mathrm{PF}^A_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$.

Then there exists W such that the following occur.

(a) $W \in \text{co-NP} \cap \mathrm{DTIME}(n^{\mathrm{O}(1)}2^{f(n)})$.

(b) *The set* W, *together with the function* $k(m)$, *satisfy the premise of Theorem* 4.4.
(c) $A \in \mathrm{NP}^W/\mathrm{poly} \cap \text{co-NP}^W/\mathrm{poly} \subseteq \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$. *If* A *is self-reducible then* $A \in \Sigma_4^p \cap \Pi_4^p$.
(d) $A \in \mathrm{DTIME}(n^{\mathrm{O}(k(n))}2^{f(n)})/\mathrm{poly}$.
(e) *If* f *is* $\mathrm{O}(\log n)$ *then* $W \in \mathrm{P}$ *so* $A \in \mathrm{NP}/\mathrm{poly} \cap \text{co-NP}/\mathrm{poly}$. *If, in addition,* A *is self-reducible then* $A \in \Sigma_3^p \cap \Pi_3^p$.

Proof. We define W so that a, b hold. Items c, d, e will follow from a, b, Theorem 4.4, and Corollary 4.6.

We define an auxiliary function q as follows. On input $z \in \Sigma^*$, find m such that $z = x_1 x_2 \cdots x_{k(m)}$ and, for all i, $|x_i| = m$ (if no such m exists then $q(z) = 0$). Let $q(z) = C^A_{f(mk(m))}(x_1, \ldots, x_{f(mk(m))})$ (this is where we use $f(mk(m)) \leqslant k(m)$). Note that on input of length $mk(m)$ (any m), q can be computed with $\leqslant f(mk(m))$ parallel queries to A, and on inputs of any other length no queries are needed. Hence $q \in \mathrm{PF}^A_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}$. By Fact 2.16 q is $2^{f(n)-1}$ enumerable. Let $e(z, -) \in \mathrm{PF}$ be the function that enumerates (in the sense of Definition 2.10) at most $2^{f(|z|)-1} < 2^{f(|z|)}$ possibilities for $q(z)$.

We define W to be the union over m of the set of ordered pairs $(t, \vec{b})$ such that the following hold.
(i) $t = \langle x_1, \ldots, x_{k(m)} \rangle$, $\vec{b} = b_1 b_2 \cdots b_{k(m)}$ where $(\forall i)[|x_i| = m$ and $b_i \in \{0, 1\}]$.
(ii) Let $z = x_1 \cdots x_{k(m)}$. $(\forall i < 2^{f(mk(m))})[e(z, i) \neq b_1 b_2 \cdots b_{f(mk(m))}]$.

We show that $k(m)$ and W satisfy the premise of Theorem 4.4. Let $t = \langle x_1, \ldots, x_{k(m)} \rangle$ and $z = x_1 \cdots x_{k(m)}$. Since $e(z, -)$ enumerates at most $2^{f(mk(m))-1} < 2^{f(mk(m))}$ possibilities, there exists $\vec{b}$ such that $(t, \vec{b}) \in W$. Hence the first premise of Theorem 4.4 is satisfied. If $(t, \vec{b}) \in W$ then, by the definition of W, the first $f(mk(m))$ bits of $\vec{b}$ differ from the first $f(mk(m))$ bits of $C^A_{k(m)}$. Hence $C^A_{k(m)}(t) \neq \vec{b}$, so the second premise of Theorem 4.4 is satisfied.

It is easy to see that $W \in \mathrm{DTIME}(n^{\mathrm{O}(1)}2^{f(n)})$ and $W \in \text{co-NP}$. □

Krentel [45] proved (assuming $\mathrm{P} \neq \mathrm{NP}$) that if $f \in \mathrm{PF}$, f is non-decreasing, and $f(n) \leqslant (1 - \epsilon)\log n$, then $\mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{T}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}$ for any X. (As noted before, Krentel actually showed this just for $f(n) \leqslant \frac{1}{2}\log n$, but his proof can be modified to $f(n) \leqslant (1 - \epsilon)\log n$. See [10].)

Corollary 5.2 (ii) extends his result.

Corollary 5.2. *Let* $f \in \mathrm{PF}$, f *be non-decreasing, and* $f(n) = \mathrm{O}(\log n)$.
(i) *If* $(\exists X)[\mathrm{PF}^A_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$, *then* $A \in \mathrm{NP}/\mathrm{poly} \cap \text{co-NP}/\mathrm{poly}$ *and* $A \in \mathrm{DTIME}(n^{\mathrm{O}(\log n)})/\mathrm{poly}$. *If* A *is self-reducible then* $A \in \Sigma_3^p \cap \Pi_3^p$.
(ii) *Let* $i \geqslant 1$. *If* $\Sigma_3^p \neq \Pi_3^p$ *then* $(\forall X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$. (*In particular* $(\forall X)[\mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$.)
(iii) *If* $\mathrm{SAT} \notin \mathrm{DTIME}(n^{\mathrm{O}(\log n)})/\mathrm{poly}$ *then* $(\forall X)[\mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$.
(iv) *If* $\Sigma_3^p \neq \mathrm{PSPACE}$ *then* $(\forall X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$.

Proof.
(i) Let $k(m) = \lfloor d \log m \rfloor$ where d is large enough so that $f(mk(m)) \leqslant k(m)$ (such a d exists since $k(m) = \mathrm{O}(\log m)$). By Theorem 5.1 $A \in \mathrm{NP}/\mathrm{poly} \cap \text{co-NP}/\mathrm{poly}$, $A \in \mathrm{DTIME}(n^{\mathrm{O}(\log n)})/\mathrm{poly}$, and if A is self-reducible then $A \in \Sigma_3^p \cap \Pi_3^p$.

(ii) If $(\exists X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$ then, by part i, NP $\subseteq$ co-NP/poly. Hence (by [72]) $\Sigma_3^p = \Pi_3^p$.

(iii) If $(\exists X)[\mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$ then, by part i, SAT $\in$ DTIME$(n^{\mathrm{O}(\log n)})$/poly.

(iv) If $(\exists X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$ then, by part i, QBF $\in \Sigma_3^p \cap \Pi_3^p$ hence $\Sigma_3^p =$ PSPACE. □

We are currently unable to extend Corollary 5.2 to $f(n) \neq \mathrm{O}(\log n)$. However we can prove a similar result about $f(n) = n^\epsilon$ queries to a Σ_i^p-complete oracle ($i \geqslant 3$) or a PSPACE-complete oracle.

Corollary 5.3. *Let ϵ be a positive real number such that $0 \leqslant \epsilon < 1$. Let f be a function such that $f \in$ PF, f is non-decreasing, and $f(n) \leqslant n^\epsilon$.*

(i) *If $(\exists X)[\mathrm{PF}^A_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$, then $A \in \Sigma_2^p$/poly $\cap\, \Pi_2^p$/poly. If A is self-reducible then $A \in \Sigma_4^p \cap \Pi_4^p$.*

(ii) *Let $i \geqslant 3$. If $\Sigma_4^p \neq \Pi_4^p$ then $(\forall X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$.*

(iii) *If $\Sigma_4^p \neq$ PSPACE then $(\forall X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$.*

Proof.

(i) Let $k(m) = m^{\frac{\epsilon}{1-\epsilon}}$. Then $f(mk(m)) \leqslant k(m)$. By Theorem 5.1.c $A \in \Sigma_2^p$/poly $\cap\, \Pi_2^p$/poly and if A is self-reducible then $A \in \Sigma_4^p \cap \Pi_4^p$.

(ii) If $(\exists X)[\mathrm{PF}^{\Sigma_i^p}_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$ then, by part i, $\Sigma_i^p \in \Sigma_2^p$/poly $\cap\, \Pi_2^p$/poly. By Lemma 4.5 we obtain $\Sigma_2^{p,\Sigma_3^p} \subseteq \Sigma_4^p$, hence $\Sigma_4^p = \Pi_4^p$.

(iii) If $(\exists X)[\mathrm{PF}^{\mathrm{PSPACE}}_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$ then, by part i, PSPACE $\in \Sigma_2^p$/poly $\cap\, \Pi_2^p$/poly. By [5, Theorem 4.3] $\Sigma_2^{p,\mathrm{PSPACE}} \subseteq \Sigma_4^p$. Hence PSPACE $\subseteq \Sigma_4^p$. □

In [10], using special properties of SAT, Beigel showed that $\mathrm{PF}^{\mathrm{NP}}_{f(n)-\mathrm{tt}} \not\subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}$ for any X unless NTIME$(n^{f(n^{\mathrm{O}(1)})}) =$ RTIME$(n^{f(n^{\mathrm{O}(1)})})$. As a Corollary to Theorem 5.1 we derive a result with a similar flavor, but without using any special properties of the oracle.

Corollary 5.4. *Let $f(n) = \log^{\mathrm{O}(1)} n$. If $(\exists X)[\mathrm{PF}^A_{f(n)-\mathrm{tt}} \subseteq \mathrm{PF}^X_{(f(n)-1)-\mathrm{T}}]$ then $A \in$ DTIME$(n^{\mathrm{polylog}n})$/poly.*

Proof. By assumption, $f(n) = \mathrm{O}(\log^i n)$ for some i. Let d be such that if $k(m) = d(\log^{i+1} m)$ then $(\forall m)[f(mk(m)) \leqslant k(m)]$. By Theorem 5.1.d $A \in$ DTIME$(n^{\mathrm{O}(k(n))}2^{f(n)})$/poly $\subseteq$ DTIME$(n^{\mathrm{polylog}n})$/poly. □

6. Applications: circuit complexity

We consider $\mathcal{G}$-circuits that make oracle queries. Following Wilson [69], we allow an oracle $\mathcal{G}$-circuit to query an oracle $X \notin \mathcal{G}$ by permitting the circuit to contain X-gates (which compute membership in X). The number of queries to X is the number of X-gates in the $\mathcal{G}$-circuit.

Notation 6.1. We have several conventions about how to interpret a circuit. If a circuit is intended to recognize a set then we assume that the nth circuit takes elements of Σ^n as input. If a circuit is intended to compute $C_{k(n)}^{A^{=n}}$ then we assume that the nth circuit takes elements of $(\Sigma^n)^{k(n)}$ as input. If a circuit is intended to compute a set of type W from Theorem 4.4 then we assume that the nth circuit takes as input elements of $(\Sigma^n)^{k(n)} \times \{0,1\}^{k(n)}$.

Let $k(n) \in \mathrm{PF}$. Let $A \subseteq \Sigma^*$. There exists a trivial polynomial-size constant depth circuit family $\{D_n\}_{n=1}^{\infty}$ such that D_n makes $k(n)$ queries to A and

$$(\forall n)(\forall t \in (\Sigma^n)^{k(n)})[D_n(t) = C_{k(n)}^{A^{=n}}(t)].$$

For which sets A can we compute $C_{k(n)}^{A^{=n}}$ with a small oracle circuit family with $k(n)-1$ queries to some X? We show that such an A must be easy to compute with a (non-oracle) circuit family. As a corollary we obtain an extension of a result by Cai [21] on PARITY.

Theorem 6.2. *Let $k(n) \in \mathrm{PF}$. Let $A \subseteq \Sigma^*$. If there is an $(s(n), d(n))$ oracle $\mathcal{G}$-circuit family $\{D_n\}_{n=1}^{\infty}$ such that D_n computes $C_{k(n)}^{A^{=n}}$ while making only $k(n)-1$ oracle queries to some oracle X then A is recognized by an $(n^{\mathrm{O}(k(n))}s(n), d(n)+\mathrm{O}(1))$ $\mathcal{G}$-circuit family.*

Proof. Assume the hypothesis. Equivalently, there exist, for each n, $2^{k(n)-1}$ $\mathcal{G}$-circuits family $D_n^1, \ldots, D_n^{2^{k(n)-1}}$ each having size $s(n)$ and depth $d(n)$ such that for every $t \in (\Sigma^n)^{k(n)}$ there exists $i \leqslant 2^{k(n)-1}$ such that $C_{k(n)}^{A}(t) = D_n^i(t)$. Let

$$W = \bigcup_{n=1}^{\infty} \{(t, \vec{b}) : t \in (\Sigma^n)^{k(n)}, \vec{b} \in \{0,1\}^{k(n)} \text{ and } (\forall i \leqslant 2^{k(n)-1})[\vec{b} \neq D_n^i(t)]\}.$$

It is easy to see that W can be recognized by a $(2^{k(n)}s(n), d(n)+\mathrm{O}(1))$ $\mathcal{G}$-circuit family. By Corollary 4.6 A can be recognized by an $(n^{\mathrm{O}(k(n))}s(n), d(n)+\mathrm{O}(1))$ $\mathcal{G}$-circuit family. □

A special case of the following corollary was originally proved by Cai [21].

Definition 6.3.

$$\mathrm{MOD}_m(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } x_1 + \cdots + x_k \equiv 0 \pmod m \\ 0 & \text{otherwise.} \end{cases}$$

Corollary 6.4. *Let $k(n) = n^{\mathrm{o}(1)}$, and let d be a positive integer. Let m be a positive integer divisible by a prime number p, and let q be a power of any prime number other than p. Let $\mathcal{G}$ consist of the* NOT *function, as well as* AND, OR, *and* MOD_q *functions of every arity. If there is an $(s(n), d)$ oracle $\mathcal{G}$-circuit family $\{C\}_{n=1}^{\infty}$ such that each circuit D_n computes $C_{k(n)}^{\mathrm{MOD}_m^{=n}}$ with $k(n)-1$ oracle queries to some oracle X, then $s(n) = 2^{n^{\Omega(1)}}$.*

Proof. Assume the hypothesis. Then by Theorem 6.2 MOD_m can be computed by an $(n^{\mathrm{O}(k(n))}s(n), d+\mathrm{O}(1))$ $\mathcal{G}$-circuit family. However, constant-depth $\mathcal{G}$-circuits for MOD_m require size $2^{n^{\Omega(1)}}$ [19,61] (see also [35–37,71]. Therefore $n^{k(n)}s(n) = 2^{n^{\Omega(1)}}$. Since $k(n) = n^{\mathrm{o}(1)}$ we obtain $s(n) = 2^{n^{\Omega(1)}}$. □

7. Applications: enumerability

Cai and Hemachandra [23] proved that #SAT is not n^k-enumerable unless $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$. We use Theorems 3.25 and 4.4 to prove many functions are not $f(n)$-enumerable for a variety of f (under suitable assumptions). We also obtain their result as a consequence of our theorems and [68, Theorem 4.1].

Our techniques can be used to obtain results for counting functions associated to many complexity classes, and for $\#GA$ (the number of automorphisms of a graph). This material is omitted here but can be found in the long version [3]. Results on enumerability and graph automorphism can also be found in [7].

In this section we study the following three functions.

Definition 7.1.

(i) #SAT is the function that, given a Boolean formula $\phi(x_1, \ldots, x_n)$, returns

$$|\{\vec{b} \in \{0,1\}^n : \phi(\vec{b}) = 1\}|.$$

(ii) $\#\mathrm{QBF}_i$ is the function that, given a quantified Boolean formula $\phi(x_1, \ldots, x_n)$ which (1) has n free variables, (2) starts with an $\exists$, and (3) has at most $i - 1$ alternations of quantifiers, returns

$$|\{\vec{b} \in \{0,1\}^n : \phi(\vec{b}) = 1\}|.$$

(iii) #QBF is the function that, given a quantified Boolean formula $\phi(x_1, \ldots, x_n)$ which (1) has n free variable, and (2) starts with an $\exists$, returns

$$|\{\vec{b} \in \{0,1\}^n : \phi(\vec{b}) = 1\}|.$$

Definition 7.2. We say that $g \leqslant_{\mathrm{m}}^{p} f$ if there exist $S, T \in \mathrm{PF}$ such that $g(x) = S(x, f(T(x)))$. Intuitively, T maps x to an element $T(x)$ such that $f(T(x))$ has information that allows one to compute $g(x)$, and S extracts that information. Hardness and completeness are defined accordingly. In this paper we will refer to T as the reduction and supress the role of S.

Definition 7.3. Let $g : \Sigma^* \to \mathsf{N}$. Then let

$$\mathrm{bit}_g = \{\langle x, 0^i \rangle : \text{the } i - \text{ th bit of the binary expansion of } g(x) \text{ is } 1\}.$$

Notation 7.4. A *quantified Boolean formula* is a Boolean formula where some of the variables (though not necessarily all) are quantified. When writing down a quantified Boolean formula with some variables free we will write $\phi(x_1, \ldots, x_m)$ to denote that $x_1, \ldots, x_m$ are the free variables. Note that for any $\vec{b} \in \{0,1\}^m$, $\phi(\vec{b})$ is either true or false. We denote that $\phi(\vec{b})$ is true (false) by $\phi(\vec{b}) = 1$ ($\phi(\vec{b}) = 0$).

We will be concerned with reducting many queries to g to just one query to g. The next definition defines a function g_q^+ that reports the answers to many queries to g. The next two sections present lemmas which allow you to show that, for some functions g, g_q^+ can be computed with one query to g.

Definition 7.5. Let g be any function and q be any polynomial. The function g_q^+ is defined on the set $\bigcup_{m=0}^{\infty} (\Sigma^{\leqslant m})^{q(m)}$ as $g_q^+(x_1, \ldots, x_{q(m)}) = \langle g(x_1), \ldots, g(x_{q(m)}) \rangle$.

7.1. Lemmas on functions associated with formulas

We now show that for certain functions g we have, for any polynomial q, $g_q^+ \leqslant_{\mathrm{m}}^p g$ with very little blowup in size. In particular we will be looking at #SAT, #QBF$_i$, and #QBF (see Definition 7.1).

Cai and Hemachandra [22] proved that $\#\mathrm{SAT}_q^+ \leqslant_{\mathrm{m}}^p \#\mathrm{SAT}$ though the idea is essentially due to Papadimitriou and Zachos [52]. Their reduction causes a polynomial blowup of size. We show that $\#\mathrm{QBF}_q^+ \leqslant_{\mathrm{m}}^p \#\mathrm{QBF}$ with very little blowup in size.

Lemma 7.6. *There exists $T \in \mathrm{PF}$ such that the following hold.*

(i) *T takes as input a finite sequence $\phi_1, \ldots, \phi_q$ of quantified Boolean formulas that have the same number of free variables.*

(ii) *T outputs a formula ϕ such that the following hold.*
   (a) *Knowing* $\#\mathrm{QBF}(\phi)$ *yields* $\langle \#\mathrm{QBF}(\phi_1), \ldots, \#\mathrm{QBF}(\phi_q)\rangle$.
   (b) $|\phi| \leqslant \mathrm{O}(|\langle \phi_1, \ldots, \phi_q\rangle|)$.

Proof. Given $\phi_1, \ldots, \phi_q$ we describe how to construct ϕ. Let m be the number of variables in each ϕ_i. Let the variables of ϕ_i be $x_{i1}, \ldots, x_{im}$. We assume without loss of generality that q is a power of 2.

We will first construct a formula ϕ' that satisfies $ii(a)$ but is too long to satisfy $ii(b)$. We then show how to obtain an equivalent formula that is shorter.

Let $i \to v_i$ be a bijection from $\{1, \ldots, q\}$ to $\{0,1\}^{\log q}$. Let $z_1, \ldots, z_{\log q}$ be new (free) variables. We define the expression $(\vec{z} = v_i)$ to be the monomial that is 1 iff we set z_j to the jth bit of v_i. Formally

$$(\vec{z} = v_i) = \left(\bigwedge_{v_i[j]=1} z_j \right) \wedge \left(\bigwedge_{v_i[j]=0} \neg z_j \right).$$

Let $y_{10}, \ldots, y_{1m}, y_{20}, \ldots, y_{2m}, \ldots, y_{q0}, \ldots, y_{qm}$ be new variables.

Let

$$\phi' = \bigvee_{i=1}^{q} \left[\phi_i \wedge (\vec{z} = v_i) \wedge \left(\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab} \right) \right].$$

Note that it is impossible for two disjuncts of ϕ' to be true at the same time. Hence

$$\#\mathrm{QBF}(\phi') = \sum_{i=1}^{q} \#\mathrm{QBF}\left(\phi_i \wedge (\vec{z} = v_i) \wedge \left(\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab} \right) \right).$$

Note that ϕ_i, $(\vec{z} = v_i)$, and $\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab}$ use disjoint sets of variables. Also note that the $(q-i)(m+1)$ variables in $\{y_{ab} : i+1 \leqslant a \leqslant q, 0 \leqslant b \leqslant m\}$ are not constrained. Hence

$$\#\mathrm{QBF}\left(\phi_i \wedge (\vec{z} = v_i) \wedge \left(\bigwedge_{a=1}^{i-1} \bigwedge_{b=0}^{m} y_{ab} \right) \right)$$

$$= \#\mathrm{QBF}(\phi_i) \cdot \#\mathrm{QBF}(\vec{z} = v_i) \cdot \#\mathrm{QBF}\left(\bigwedge_{a=1}^{i-1}\bigwedge_{b=0}^{m} y_{ab}\right)$$
$$= \#\mathrm{QBF}(\phi_i) \cdot 2^{(q-i)(m+1)}.$$

Putting this all together we obtain

$$\#\mathrm{QBF}(\phi') = \sum_{i=1}^{q} 2^{(q-i)(m+1)} \#\mathrm{QBF}(\phi_i).$$

Since $(\forall i)[\#\mathrm{QBF}(\phi_i) < 2^{m+1}]$ all the values $\#\mathrm{QBF}(\phi_i)$ can be easily deduced from $\#\mathrm{QBF}(\phi')$.

The formula ϕ' would be an ideal candidate for $T(\phi_1, \ldots, \phi_q)$ except that it is too long. We actually output a shorter formula that is equivalent to ϕ'. The idea is to introduce new variables $w_1, \ldots, w_q$ such that w_i will be equivalent to $\bigwedge_{a=1}^{i-1}\bigwedge_{b=0}^{m} y_{ab}$. The formula ϕ is the conjunction of the following three formulas.

(i) $w_1 = \bigwedge_{b=0}^{m} y_{1b}$.
(ii) $\bigwedge_{i=2}^{q}[w_i = (w_{i-1} \wedge \bigwedge_{b=0}^{m} y_{ib})]$.
(iii) $\bigvee_{i=1}^{q}[\phi_i \wedge (\vec{z} = v_i) \wedge w_i]$.

Clearly $\#\mathrm{QBF}(\phi) = \#\mathrm{QBF}(\phi')$. Note that

$$|\phi| = \mathrm{O}(m) + \mathrm{O}(mq) + \sum_{i=1}^{q} \mathrm{O}(|\phi_i| + \log m) = \mathrm{O}(|\langle \phi_1, \ldots, \phi_q \rangle|). \qquad \square$$

Definition 7.7.

Let A be a set of quantified Boolean formulas. The set A is *nice* if the following hold.

(i) All Boolean formulas (without quantifiers) are in A.
(ii) If $\phi_1, \phi_2 \in A$ then $\phi_1 \vee \phi_2 \in A$ and $\phi_1 \wedge \phi_2 \in A$.

For a set of formulas to be nice we do *not* require that it be a minimal set of formulas that satisfy the conditions. For example the set of formulas that have at most i alternations of quantifiers is nice.

Lemma 7.8. *Let A be a nice set of quantified Boolean formulas. Let f be the function* #QBF *restricted to A. For all polynomials q, $f_q^+ \leqslant_{\mathrm{m}}^p f$ via some reduction T_q where $(\forall z)[|T_q(z)| = \mathrm{O}(|z|^{1+\frac{1}{\deg(q)}})]$.*

Proof. Fix a polynomial q. Let an input to f_q^+ be $\langle \phi_1, \ldots, \phi_{q(m)} \rangle$ where $(\forall i)[|\phi_i| \leqslant m]$. Assume that ϕ_i has $m_i \leqslant m$ variables. Let $\psi_i = \phi_i \wedge \bigwedge_{b=m_i+1}^{m} x_{ib}$ where the x_{ib} variables are new free variables. Note that for each i (1) ψ_i has exactly m variables, (2) $f(\phi_i) = f(\psi_i)$, and (3) $|\psi| = |\phi_i| + \mathrm{O}(m)$.

The vector $\langle \psi_1, \ldots, \psi_{q(m)} \rangle$ is in the domain of the transformation T from Lemma 7.6. Let

$$T_q(\langle \phi_1, \ldots, \phi_{q(m)} \rangle) = T(\langle \psi_1, \ldots, \psi_{q(m)} \rangle).$$

Since $T(\langle \psi_1, \ldots, \psi_{q(m)} \rangle)$ yields all the $f(\psi_i)$ and $f(\psi_i) = f(\phi_i)$, clearly $T_q(\langle \phi_1, \ldots, \phi_{q(m)} \rangle)$ yields all the $f(\phi_i)$. Since A is nice, clearly $T_q(\langle \phi_1, \ldots, \phi_{q(m)} \rangle) \in A$.

Note that

$$\begin{aligned}|T_q(\langle\phi_1,\ldots,\phi_{q(m)}\rangle)| &= |T(\langle\psi_1,\ldots,\psi_{q(m)}\rangle)|\\ &= \mathrm{O}\left(|\langle\psi_1,\ldots,\psi_{q(m)}\rangle|\right)\\ &= \mathrm{O}\left(\sum_{i=1}^{q(m)}|\psi_i|\right)\\ &= \mathrm{O}\left(\sum_{i=1}^{q(m)}|\phi_i|+m\right)\\ &= \mathrm{O}\left(\sum_{i=1}^{q(m)}2m\right)\\ &= \mathrm{O}(mq(m))\\ &= \mathrm{O}(q(m)^{1+\frac{1}{\deg(q)}}).\end{aligned}$$

Since the length of the input is $\Omega(q(m))$ and the length of the output is $\mathrm{O}(q(m)^{1+\frac{1}{\deg(q)}})$, we are done. □

Lemma 7.9. *Let f be* #SAT, #QBF$_i$, *or* #QBF. *For all polynomials q, $f_q^+ \leqslant_{\mathrm{m}}^p f$ via some reduction T_q where $(\forall z)[|T_q(z)| = \mathrm{O}(|z|^{1+\frac{1}{\deg(q)}})]$.*

7.2. $b(n)$-Enumerable for large $b(n)$

The main theorem of this section establishes conditions on a function g and a real $0 < \epsilon < 1$ that cause the implication

$$g \text{ is } 2^{n^\epsilon}\text{-enumerable} \Rightarrow \mathrm{P}^g \subseteq \Sigma_4^p \cap \Pi_4^p.$$

We will apply this to #SAT, #QBF$_i$, and #QBF. In addition we show that if any #P-hard function is n^k-enumerable then $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$.

Theorem 7.10. *Let b and g be functions and A be a set such that the following hold.*

(p1) *There exists a polynomially bounded function q such that $C_q^A \leqslant_{\mathrm{m}}^p g$ via a reduction T such that $(\overset{\infty}{\forall} m)(\forall t \in (\Sigma^{\leqslant m})^{q(m)})[b(|T(t)|) \leqslant 2^{q(m)} - 1]$.($t$ is the code for a $q(m)$-tuple of strings, each of which is $\leqslant m$ in length. Formally it is of the form $x_1\%x_2\%\cdots\%x_{q(m)}$ where each x_i has length $\leqslant m$.)*

(p2) *g is $b(n)$-enumerable.($b(n)$ need not be bounded by a polynomial so we use the definition of enumerable stated in Definition* 2.10.)

(p3) *A is self-reducible.*

Then the following occur.

(a) *$A \in \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}$ and $\mathrm{P}^A \subseteq \Sigma_4^p \cap \Pi_4^p$.*

(b) *If* $A = \mathrm{bit}_g$ *then* $\mathrm{P}^g \subseteq \Sigma_4^p \cap \Pi_4^p$. *In this case the function* g_q^+ *can replace* C_q^A *in premise* $p1$.
(c) *If the function* q *in premise* $p1$ *is such that* $q(m) = \mathrm{O}(\log m)$ *then the conclusion can be strengthened to* $A \in \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$ *and* $\mathrm{P}^A \subseteq \Sigma_3^p \cap \Pi_3^p$. *If in addition* $A = \mathrm{bit}_g$ *then* $\mathrm{P}^g \subseteq \Sigma_3^p \cap \Pi_3^p$. *In this case the function* g_q^+ *can replace* C_q^A *in premise* $p1$.

Proof.

(a). We will find a set $W \in \Pi_1^p$ such that A satisfies the premise of Theorem 4.4 and Corollary 4.6.c. with W and q. By Theorem 4.4 we will have

$$A \in \mathrm{NP}^W/\mathrm{poly} \cap \mathrm{co\text{-}NP}^W/\mathrm{poly} \subseteq \Sigma_2^p/\mathrm{poly} \cap \Pi_2^p/\mathrm{poly}.$$

By Corollary 4.6 we will have $\mathrm{P}^A \subseteq \Sigma_4^p \cap \Pi_4^p$.

To define W we will need a function e. We state what properties e will need, define W, and then show that such an e exists.

The function e will have the following properties.

1. $e \in \mathrm{PF}$.
2. $e : \bigcup_{m=0}^{\infty} (\Sigma^m)^{q(m)} \times \mathsf{N} \to \Sigma^*$.
3. $(\overset{\infty}{\forall} m)(\forall t \in (\Sigma^m)^{q(m)})(\exists i \leqslant 2^{q(m)} - 2)[e(t,i) = C_{q(m)}^A(t)]$.

Once we have e we can define W as follows. Let W be the union over m of the set of ordered pairs $(t, \vec{c})$ where the following hold.

(a) $t \in (\Sigma^m)^{q(m)}$ and $\vec{c} \in \{0,1\}^{q(m)}$.
(b) $(\forall i \leqslant 2^{q(m)} - 2)[\vec{c} \neq e(t,i)]$. ($W$ will be non-trivial since this condition only eliminates $\leqslant 2^{q(m)} - 1$ choices for $\vec{c}$, namely $e(t,0), \ldots, e(t, 2^{q(m)} - 2)$.)

It is easy to see that A satisfies the premise of Theorem 4.4 with this choice of W and q and that $W \in \Pi_1^p$.

It remains to define e. Since g is $b(n)$-enumerable there exists a function $e' \in \mathrm{PF}$, $e' : \Sigma^* \times \mathsf{N} \to \Sigma^*$, that $b(n)$-enumerates g (in the sense of Definition 2.10). We will use e' later.

Let $m \in \mathsf{N}$, and $t = x_1 \% \cdots \% x_{q(m)}$ (where $(\forall i)[|x_i| = m]$). Note that $t \in (\Sigma^{\leqslant m} \%)^{q(m)-1} \Sigma^{\leqslant w}$. Let $i \in \mathsf{N}$. We describe how to compute $e(t,i)$.

First compute $z = T(t)$. Every possibility for $g(z)$ yields a possibility for $C_q^A(t)$. Let $POSS \in \mathrm{PF}$ map possibilities for $g(z)$ to the corresponding possibilities for $C_q^A(t)$. Let $e(t,i) = POSS(e'(z,i))$. Clearly $e \in \mathrm{PF}$.

By the definition of e' we know that $(\exists i \leqslant b(|z|) - 1)[e'(z,i) = g(z)]$, hence $(\exists i \leqslant b(|T(t)|))[e(t,i) = C_{q(m)}^A(t)]$. Since $t \in \Sigma^{\leqslant m} \%^{q(m)}$ premise $p1$ yields $(\overset{\infty}{\forall} m)[b(|T(t)|) - 1 \leqslant 2^{q(m)} - 2]$. Hence stipulation 3 on the function e is met.

(b) $\mathrm{P}^g \subseteq \mathrm{P}^{\mathrm{bit}_g} = \mathrm{P}^A \subseteq \Sigma_4^p \cap \Pi_4^p$.
(c) If $q(m) = \mathrm{O}(\log m)$ then $W \in \mathrm{P}$ hence $A \in \mathrm{NP}^W/\mathrm{poly} \cap \mathrm{co\text{-}NP}^W/\mathrm{poly} = \mathrm{NP/poly} \cap \mathrm{co\text{-}NP/poly}$. Since A is self-reducible, by Corollary 4.6 $\mathrm{P}^A \subseteq \Sigma_3^p \cap \Pi_3^p$. If $A = \mathrm{bit}_g$ then $\mathrm{P}^g \subseteq \mathrm{P}^{\mathrm{bit}_g} = \mathrm{P}^A \subseteq \Sigma_3^p \cap \Pi_3^p$. □

Corollary 7.11. *Let* ϵ *be any real such that* $0 \leqslant \epsilon < 1$. *Let* $i \geqslant 1$.
(i) *If* #SAT *is* 2^{n^ϵ}*-enumerable then* $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.
(ii) *If* $\#\mathrm{QBF}_i$ *is* 2^{n^ϵ}*-enumerable then* $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

(iii) *If* #QBF *is* 2^{n^ϵ}*-enumerable then* $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{PSPACE}$.

Proof. We prove (i). Let $g = \#\mathrm{SAT}$. We assume that g is 2^{n^ϵ}-enumerable. We will show that the premise of Theorem 7.10 (the version stated in conclusion b with g_q^+ instead of C_q^A) is satisfied with $b(n) = 2^{n^\epsilon}$, g, and $A = \mathrm{bit}_g$. This will yield $\mathrm{P}^g \subseteq \Sigma_4^p \cap \Pi_4^p$ and hence $\mathrm{P}^{\#\mathrm{P}} \subseteq \Sigma_4^p \cap \Pi_4^p$. By [67] $\Sigma_4^p \cap \Pi_4^p \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$, hence $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

Clearly premises $p2$ and $p3$ of Theorem 7.10 are satisfied. We show that premise $p1$ is satisfied. Let $q(m) = m^\alpha$ where α will be specified later. By Lemma 7.9 $g_q^+ \leqslant_{\mathrm{m}}^p g$ via a reduction T such that $(\forall z)[|T(z)| \leqslant \mathrm{O}(|z|^{1+\frac{1}{\alpha}})]$. In order to apply Theorem 7.10 we need that $(\alpha+1)(1+\frac{1}{\alpha})\epsilon < \alpha$. Since $\epsilon < 1$ there exists a large enough α to make this inequality true.

The proof of (ii) is similar to the proof of (i). A proof similar to (i) can establish that if the premise of (iii) holds then $\Sigma_4^p \cap \Pi_4^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{PSPACE}}$. However, it is easy to see that any function in #PSPACE can be computed with polynomial space. Hence $\mathrm{P}^{\#\mathrm{PSPACE}} = \mathrm{P}^{\mathrm{PSPACE}} = \mathrm{PSPACE}$. □

The following corollary has been obtained independently by Cai and Hemachandra [23].

Corollary 7.12. *If* #SAT *is* $p(n)$*-enumerable for some polynomial* p *then* $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$.

Proof. If #SAT is $p(n)$-enumerable then there exists $\epsilon > 0$ such that #SAT is 2^{n^ϵ}-enumerable. Hence, by Corollary 7.11 $\mathrm{P}^{\#\mathrm{P}} = \Sigma_4^p \cap \Pi_4^p$.

We show that P = NP which will imply $\mathrm{P} = \Sigma_4^p$. Combining that with the above yields $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$.

Let MAXSAT be the following problem: given a CNF-formula, find a truth assignment that maximizes the number of clauses that are satisfied. If there is more than one then output the lexicgraphically least such truth assignment. Krentel [45] showed that $\mathrm{MAXSAT} \in \mathrm{PF}^{\mathrm{SAT}}$ via binary search techniques. Toda and Watanabe [68, Theorem 4.1] showed that $\mathrm{PF}^{\mathrm{PH}} \subseteq \mathrm{PF}^{\#\mathrm{P}[1]}$, hence $\mathrm{MAXSAT} \in \mathrm{PF}^{\#\mathrm{P}[1]}$. Since #SAT is $\leqslant_{\mathrm{m}}^p$-hard for #P and #SAT is $p(n)$-enumerable we know MAXSAT is $q(n)$-enumerable for some polynomial q. We use this to show $\mathrm{SAT} \in \mathrm{P}$.

Given a formula ϕ we enumerate $q(n)$ possible values of $\mathrm{MAXSAT}(\phi)$. Since there are only $q(n)$ of them we can plug each one into ϕ. If any of them satisfy ϕ then $\phi \in \mathrm{SAT}$, otherwise $\phi \notin \mathrm{SAT}$. □

All the theorems and corollaries easily relativize (using a relativized version of Theorem 4.4). From these relativized results we obtain the following corollaries.

Definition 7.13. Let $b(n)$ be a function with range N. Let $\mathcal{A} \subseteq 2^{\Sigma^*}$ and f be a function. The function f is $(b(n), \mathcal{A})$*-enumerable* if there exist $A \in \mathcal{A}$ and $e \in \mathrm{PF}^A$ such that $e : \Sigma^* \times \mathsf{N} \to \Sigma^*$ and $(\forall x)(\exists i < b(|x|)[e(x,i) = f(x)]$. (We need to have $i < b(|x|)$ instead of $i \leqslant b(|x|)$ since the natural numbers N include 0. We assume the second input to e is written in binary.)

Corollary 7.14. *Let* $i, j \geqslant 1$. *Let* ϵ *be any real such that* $0 \leqslant \epsilon < 1$.

(i) *If* #SAT *is* $(2^{n^\epsilon}, \Sigma_j^p)$*-enumerable then* $\Sigma_{j+4}^p \cap \Pi_{j+4}^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

(ii) *If* $\#\mathrm{QBF}_i$ *is* $(2^{n^\epsilon}, \Sigma_j^p)$*-enumerable then* $\Sigma_{j+4}^p \cap \Pi_{j+4}^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{P}}$.

(iii) *If* #QBF *is* $(2^{n^\epsilon}, \Sigma_j^p)$*-enumerable then* $\Sigma_{j+4}^p \cap \Pi_{j+4}^p = \mathrm{PH} = \mathrm{P}^{\#\mathrm{PSPACE}}$.

It is open whether the following is true: If #SAT is 2^{n^ϵ}-enumerable then P = NP. Stephan [63] has shown that for every superpolynomial f there is a relativized world such that #SAT is $f(n)$-enumerable and P $\neq$ NP. Since our techniques all relativize it is unlikely that they will suffice to solve the open question.

8. Structural properties

For brevity we will omit most proofs in this section. Complete details can be found in [3].
In order to express some of our results, we require a restricted version of p-superterseness.

Definition 8.1. A set is k-p-superterse if $(\forall X)[C_k^A \notin \mathrm{PF}^X_{(k-1)-\mathrm{T}}]$.

8.1. Closure properties for cheatable sets

By Lemma 3.20 the class of cheatable sets is closed under $\leqslant^p_{\mathrm{T}}$-reductions. In fact, the class of cheatable sets is also closed under union, intersection, and join.

Theorem 8.2. *If A is i-cheatable and B is j-cheatable then $A \cap B$, $A \cup B$, and $A \oplus B$ are $(i + j)$-cheatable.*

Proof. This follows from Lemma 3.20 as an easy exercise. □

As the next theorem shows, the result above is tight. Thus union, intersection, and join cause a loss of some cheatability, while Turing reductions preserve k-cheatability exactly (Lemma 3.20).

Theorem 8.3. *There exist sets A and B such that A is i-cheatable and B is j-cheatable, but $A \cap B$ and $A \oplus B$ are not $(i + j - 1)$-cheatable. (In fact, both are $(i + j)$-p-superterse.)*

Proof. A supersparse set with the desired properties can be given by a straightforward diagonalizaiton. □

Note. Let $C = \overline{A}$ and $D = \overline{B}$ where A and B are the sets constructed in Theorem 8.3. Since k-cheatability is preserved under complement, C is i-cheatable, D is j-cheatable and $C \cup D$ is not $(i + j - 1)$-cheatable. Thus a version of Theorem 8.3 for unions holds as well.

8.2. p-Selective sets

In this section we show that cheatability and p-selectivity are incomparable.

Theorem 8.4. *The following exist.*
(i) 1-*cheatable sets that are p-selective but not in* P.
(ii) 1-*cheatable sets that are not p-selective.*
(iii) *Non-cheatable sets that are p-selective.*
(iv) *Non-cheatable sets that are not p-selective.*

Proof. In [4] tally sets are constructed that are 1-cheatable but not in P. Let T_1 be such a set. It is easy to construct tally sets that are not cheatable (e.g., non-recursive tally sets [12]). Let T_2 be such a set. In [59], Selman shows that given any tally set $T \notin \mathrm{P}$ there are sets A, $B \equiv_{\mathrm{T}}^{p} T$ such that A is p-selective and B is not p-selective. (Let A be the set of strings lexicographically preceding the characteristic sequence of T. Let $B = T \oplus \overline{T}$.) For $i = 1, 2$ let A_i and B_i be such that A_i, $B_i \equiv_T^p T_i$, A_i is p-selective and B_i is not p-selective. Since k-cheatability is preserved by polynomial-time Turing equivalence A_1 is 1-cheatable and p-selective, B_1 is 1-cheatable and not p-selective, A_2 is non-cheatable and p-selective, and B_2 is non-cheatable and not p-selective. □

8.3. p-Superterse degrees

Definition 8.5. Let $\leqslant_r^p$ denote a polynomial-time reducibility. An $\leqslant_r^p$*-degree* is an equivalence class of the relation $\equiv_r^p$. We say that a degree is cheatable, p-superterse, etc., if it contains a set that is respectively cheatable, p-superterse, etc.

Theorem 8.6. *Let* **d** *be a* $\leqslant_{\mathrm{tt}}^{p}$*- or* $\leqslant_{\mathrm{T}}^{p}$*-degree. Then* **d** *is p-superterse iff* **d** *is not cheatable.*

Proof. Let D be any set in **d** and let A be $\leqslant_{\mathrm{m}}^{p}$-complete for $\mathrm{P}_{\mathrm{tt}}^{D}$ (e.g., take $A = D^{\mathrm{tt}}$, see [54,62]). We claim that A is either p-superterse or cheatable. Suppose that A is not p-superterse, so there exist k and X such that $C_k^A \in \mathrm{PF}_{(k-1)-\mathrm{T}}^{X}$. By Fact 2.17(iii). there exists $B \in \mathrm{P}_{k-\mathrm{tt}}^{A} \subseteq \mathrm{P}_{\mathrm{tt}}^{D}$ such that $C_k^A \in \mathrm{PF}_{(k-1)-\mathrm{tt}}^{B}$. Since $B \in \mathrm{P}_{\mathrm{tt}}^{D}$, we have $B \leqslant_{\mathrm{m}}^{p} A$. Therefore $C_k^A \in \mathrm{PF}_{(k-1)-\mathrm{tt}}^{A}$, so A is k-cheatable by [11, Observation 6.2]. □

Lemma 8.7. *Let A be a set.*

(i) *If A is k-cheatable then every* $B \equiv_{\mathrm{T}}^{p} A$ *is k-cheatable.*

(ii) *If A is not k-cheatable then there exists a k-p-superterse set* $B \equiv_{2^k-\mathrm{tt}}^{p} A$.

Proof.

(i) Assume A is k-cheatable. By Lemma 3.20 every set $B \leqslant_{\mathrm{T}}^{p} A$ is k-cheatable.

(ii) Assume A is not k-cheatable. Then $(\forall X)[C_{2^k}^A \notin \mathrm{PF}_{k-\mathrm{T}}^{X}]$. Find the maximum $a < 2^k$ such that $(\exists X)[C_a^A \in \mathrm{PF}_{k-\mathrm{T}}^{X}]$. By Fact 2.17(iii) there exists $B \equiv_{2^k-\mathrm{tt}}^{p} A$ such that $C_a^A \in \mathrm{PF}_{k-\mathrm{tt}}^{B}$. We prove by contradiction that B is k-p-superterse. If B is not k-p-superterse then $C_k^B \in \mathrm{PF}_{(k-1)-\mathrm{T}}^{Z}$ for some Z, so $C_a^A \in \mathrm{PF}_{(k-1)-\mathrm{T}}^{Z}$. Hence $C_{a+1}^A \in \mathrm{PF}_{k-\mathrm{T}}^{Z\oplus A}$. This contradicts the maximality of a. □

Corollary 8.8. *Let* **d** *be a* $\leqslant_{\mathrm{tt}}^{p}$*- or* $\leqslant_{\mathrm{T}}^{p}$*-degree. Then* **d** *is k-p-superterse iff* **d** *is not k-cheatable.*

Acknowledgments

We thank Eric Allender for pointing out that we obtain membership in EL_2 in Theorem 3.8, Pankaj Rohatgi for the current form of Theorem 4.4, Carsten Lund for help on Section 7, Frank Stephan for help with some of the generalizations, and Seinosuke Toda for making the observation that led to Corollary

7.12. We also thank José Balcázar Sandeep Bhatt, Michael Fisher, Albrecht Hoene, Steve Homer, Anna Karlin, Rao Kosaraju, Martin Kummer, Alan Selman, and Leen Torevliet for helpful discussions. We would like to thank Jorge Castro, Richard Chang, James Foster, James Glenn, Georgia Martin, Carlos Seara, Todd Wareham, and Yao Yong for proofreading and commentary.

Appendix A

A.1 A variant on the **BP** *operator*

Schöning [58] defined the **BP** operator as a generalization of the complexity class BPP. We will need a generalization $\mathbf{BP}^\bullet$ of the **BP** operator. Our approach closely follows his; hence we provide only sketches.

Recall Schöning's definition of the **BP** operator.

Definition A.1. Let $\mathcal{C}$ be a class of sets. We say that $A \in \mathbf{BP} \cdot \mathcal{C}$ iff there exists a set $B \in \mathcal{C}$ and a polynomial p such that for all $n \in \mathsf{N}$, for all $x \in \Sigma^n$,

$$x \in A \Rightarrow \Pr[\langle x, y\rangle \in B : y \in \Sigma^{p(n)} \text{ uniformly }] \geqslant \frac{3}{4},$$
$$x \notin A \Rightarrow \Pr[\langle x, y\rangle \notin B : y \in \Sigma^{p(n)} \text{ uniformly }] \geqslant \frac{3}{4}.$$

In Schöning's definition the string y ranged over all of $\Sigma^{p(n)}$. We need to consider what happens if y ranges over some subset $Y \subseteq \Sigma^{p(n)}$.

Definition A.2. Let $\mathcal{C}$ be a class of sets. We say that $A \in \mathbf{BP}^\bullet \cdot \mathcal{C}$ iff there exists a set $B \in \mathcal{C}$, a polynomial p, and a set $Y \subseteq \Sigma^*$ such that for all $n \in \mathsf{N}$, for all $x \in \Sigma^n$,

$$x \in A \Rightarrow \Pr[\langle x, y\rangle \in B : y \in Y \cap \Sigma^{p(n)} \text{ uniformly }] \geqslant \frac{3}{4},$$
$$x \notin A \Rightarrow \Pr[\langle x, y\rangle \notin B : y \in Y \cap \Sigma^{p(n)} \text{ uniformly }] \geqslant \frac{3}{4}.$$

Definition A.3. Let A, B be sets. We say that $A \leqslant^p_{\text{pos}} B$ iff $A \leqslant^p_{\text{T}} B$ via an oracle Turing machine $M^{()}$ with the additional property that $(\forall X, Y)[X \subseteq Y \Rightarrow L(M^X) \subseteq L(M^Y)]$. A class of sets $\mathcal{C}$ is *closed under positive reductions* if for all A, B, if $B \in \mathcal{C}$ and $A \leqslant^p_{\text{pos}} B$ then $A \in \mathcal{C}$.

Lemma A.4. *Let $\mathcal{C}$ be a class of sets closed under positive reductions. For any set $A \in \mathbf{BP}^\bullet \cdot \mathcal{C}$ and any polynomial q there is a set $B \in \mathcal{C}$, a polynomial p, and a set $Y \subseteq \Sigma^*$ such that for all $n \in \mathsf{N}$, for all $x \in \Sigma^n$,*

$$x \in A \Rightarrow \Pr[\langle x, y\rangle \in B : y \in Y \cap \Sigma^{p(n)}] \geqslant 1 - \frac{1}{2^{q(n)}},$$
$$x \notin A \Rightarrow \Pr[\langle x, y\rangle \notin B : y \in Y \cap \Sigma^{p(n)}] \geqslant 1 - \frac{1}{2^{q(n)}}.$$

Proof. Since $A \in \mathbf{BP}^{\bullet} \cdot \mathcal{C}$ there exists a set $B' \in \mathcal{C}$, a polynomial p' and a set $Y' \subseteq \Sigma^*$ such that for all $x \in \Sigma^n$,

$$x \in A \Rightarrow \Pr[\langle x, y\rangle \in B' : y \in Y' \cap \Sigma^{p'(n)}] \geqslant \frac{3}{4},$$
$$x \notin A \Rightarrow \Pr[\langle x, y\rangle \notin B' : y \in Y' \cap \Sigma^{p'(n)}] \geqslant \frac{3}{4}.$$

Let

$$t(n) = \left\lceil \frac{2}{\log 4/3} q(n) \right\rceil,$$
$$Y = \bigcup_{n=0}^{\infty} \{\langle y_1, \ldots, y_{t(n)}\rangle : (\forall i)[|y_i| = p'(n) \text{ and } y_i \in Y']\},$$
$$B = \bigcup_{n=0}^{\infty} \{\langle x, \langle y_1, \ldots, y_{t(n)}\rangle\rangle : \text{ a majority of the } \langle x, y_i\rangle \text{ are in } B'\}.$$

Let $p(n)$ be the length of $\langle y_1, \ldots, y_{t(n)}\rangle$ where all the y_i are of length $p'(n)$ (this $p(n)$ will depend on $t(n)$ and the pairing function being used).

Note that $B \in \mathcal{C}$ since $\mathcal{C}$ is closed under positive reductions. The rest of the proof proceeds exactly like [58, Lemma 3.3]. □

Theorem A.5. *If $\mathcal{C}$ is closed under positive reductions then* $\mathbf{BP}^{\bullet} \cdot \mathcal{C} \subseteq \mathcal{C}/\text{poly}$.

Proof. Let $A \in \mathbf{BP}^{\bullet} \cdot \mathcal{C}$. Apply Lemma A.4 with $q(n) = n + 1$ to obtain $B \in \mathcal{C}$, a polynomial p, and $Y \subseteq \Sigma^*$ such that

$$x \in A \Rightarrow \Pr[\langle x, y\rangle \in B : y \in Y \cap \Sigma^{p(n)}] \geqslant 1 - \frac{1}{2^{n+1}},$$
$$x \notin A \Rightarrow \Pr[\langle x, y\rangle \in B : y \notin Y \cap \Sigma^{p(n)}] \geqslant 1 - \frac{1}{2^{n+1}}.$$

By a probabilistic argument we show that, for all n, there exists a $y \in Y \cap \Sigma^{p(n)}$ such that $(\forall x)[x \in A \text{ iff } B(x, y)]$.

$$\begin{aligned}
&\Pr[(\exists x \in \Sigma^n)[B(x, y) \neq A(x)] : y \in Y \cap \Sigma^{p(n)}] \\
&\quad \leqslant \sum_{x \in \Sigma^n} \Pr[B(x, y) \neq A(x) : y \in Y \cap \Sigma^{p(n)}] \\
&\quad \leqslant (|\Sigma|)^n \cdot \frac{1}{2^{n+1}} \\
&\quad \leqslant \frac{1}{2}.
\end{aligned}$$

Hence at least one such y must exist. That string y can serve as advice. □

Definition A.2 requires that its given condition hold for all n. However, we will be dealing with languages where this condition only holds for some n, and for those n, we still need polynomial advice.

Theorem A.6. *Let $\mathcal{C}$ be any class of languages closed under positive reductions. Let $I \subseteq \mathsf{N}$. Let A be a set such that there exist $B \in \mathcal{C}$, a polynomial p, a set $Y \subseteq \Sigma^*$ such that for all $n \in I$, for all $x \in \Sigma^n$,*

$$x \in A \Rightarrow \Pr[\langle x, y\rangle \in B : y \in Y \cap \Sigma^{p(n)}] \geqslant \frac{3}{4},$$
$$x \notin A \Rightarrow \Pr[\langle x, y\rangle \notin B : y \in Y \cap \Sigma^{p(n)}] \geqslant \frac{3}{4}.$$

Then $A' = A \cap \bigcup_{n\in I} \Sigma^n \in \mathcal{C}/\text{poly}$.

Proof. For each length n one bit of advice tells if $n \in I$ or not. If $n \in I$ then proceed as in Theorem A.5, using an analogue of Lemma A.4. If $n \notin I$ then clearly $A' \cap \Sigma^n = \emptyset$. □

Appendix B

B.1 A variant of the Ajtai and Ben-Or construction

Ajtai and Ben-Or [2] showed how to convert a probabilistic circuit into a deterministic circuit with only a constant increase in depth and a polynomial increase in size. We need a variant of their theorem. Our approach closely follows theirs; hence we provide only sketches.

Definition B.1. Let C be a circuit on n inputs. The circuit C *separates* A from B if

$$x \in A \Rightarrow C(x) = 1$$
$$x \in B \Rightarrow C(x) = 0.$$

Recall the standard definition of a probabilistic circuit.

Definition B.2. The circuit model allows negation, unbounded fan-in and-gates, and unbounded fan-in or-gates. The inputs are $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_m\}$ and their negations. The x_i are called *input variables* and the y_i are called *random variables*. Let $0 \leqslant q \leqslant p \leqslant 1$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let C be a probabilistic circuit on n inputs and m random variables. The circuit C *separates A from B with probabilities* (p, q), denoted by (C, A, B, p, q), if

$$x \in A \Rightarrow \Pr[C(x, y) = 1 : y \in \{0,1\}^m \text{ uniformly }] \geqslant p,$$
$$x \in B \Rightarrow \Pr[C(x, y) = 1 : y \in \{0,1\}^m \text{ uniformly }] \leqslant q.$$

In this definition the string y ranged over all of $\{0,1\}^m$. We need to consider what happens if y ranges over some subset $Y \subseteq \{0,1\}^m$.

Definition B.3. Let $0 \leqslant q \leqslant p \leqslant 1$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let C be a probabilistic circuit on n inputs and m random variables. Let $Y \subseteq \{0,1\}^m$. The circuit C *separates A from B with probabilities* (p, q) *using* Y (denoted $\text{SEP}(C, A, B, p, q, Y)$) if for all $x \in \Sigma^n$,

$$x \in A \Rightarrow \Pr[C(x, y) = 1 : y \in Y \cap \{0,1\}^m \text{ uniformly}] \geqslant p,$$
$$x \in B \Rightarrow \Pr[C(x, y) = 1 : y \in Y \cap \{0,1\}^m \text{ uniformly}] \leqslant q.$$

The following is Lemma 1 from [2] in our framework.

Lemma B.4. *Let $n, m \in \mathbb{N}$ and $0 \leqslant q \leqslant p \leqslant 1$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let $Y \subseteq \{0,1\}^m$. Let C be a probabilistic circuit having size s and depth d, such that* $\mathrm{SEP}(C, A, B, p, q, Y)$.

(i) *If $p \geqslant p_1$ and $q \leqslant q_1$ then* $\mathrm{SEP}(C, A, B, p_1, q_1, Y)$.
(ii) *There is a circuit C^b having size s and depth d such that* $\mathrm{SEP}(C^b, B, A, 1-q, 1-p, Y)$.
(iii) *For any natural number u there are a circuit C^u having size $us+1$ and depth $d+1$ and a set $Y' \subseteq \{0,1\}^m$ such that* $\mathrm{SEP}(C^u, A, B, p^u, q^u, Y')$.
(iv) *If $1 - p + q < 2^{-n}$ then there is a deterministic circuit C^d having size s and depth d that separates A from B. Hence we have* $\mathrm{SEP}(C^d, A, B, 1, 0, Y)$.

Proof.
(i) and (ii) follow from the definition.
(iii) Let C^u consist of u independent copies of C where all the outputs are connected to one $\bigwedge$ gate.
Let $Y' = Y \times \cdots \times Y$ (u times). C^u has size $us+1$ and depth $d+1$.
Since probabilities multiply, clearly $\mathrm{SEP}(C^u, A, B, p^u, q^u, Y')$.
(iv) Identical to Lemma 1.*d* of [2]. □

We now state three lemmas that can be proved from Lemma B.4. Proofs are omitted; however they are similar to the proofs of Lemmas 2 and 3 and Theorem 4 of [2].

Lemma B.5. *Let $r \geqslant 2$. Let $n, m \in \mathbb{N}$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let $Y \subseteq \{0,1\}^m$. Let C be a probabilistic circuit having size s and depth d, such that* $\mathrm{SEP}(C, A, B, \frac{1}{2}(1 + (\log n)^{-r}), \frac{1}{2}, Y)$. *Then there exists a circuit C' of size $sn^2 \log n$ and depth $d+2$ such that* $\mathrm{SEP}(C, A, B, \frac{1}{2}(1+(\log n)^{-r+1}), \frac{1}{2}, Y)$.

Lemma B.6. *Let $n, m \in \mathbb{N}$. Let A, B be disjoint subsets of $\{0,1\}^n$. Let $Y \subseteq \{0,1\}^m$. Let C be a probabilistic circuit having size s and depth d, such that* $\mathrm{SEP}(C, A, B, \frac{1}{2}(1 + (\log n)^{-1}), \frac{1}{2}, Y)$. *Then there exists a circuit C' of size sn^8 and depth $d+4$ such that* $\mathrm{SEP}(C', A, B, 1, 0, Y)$.

Lemma B.7. *Let $\{C\}_{n=1}^{\infty}$ be an $(s(n), d(n))$ probabilistic circuit family where D_n has $k(n)$ random variables. Let $\{Y\}_{n=1}^{\infty}$ be such that $Y_n \subseteq \{0,1\}^{k(n)}$. Assume that for all n,* $\mathrm{SEP}(D_n, A_n, \overline{A}_n, \frac{3}{4}, \frac{1}{4}, Y_n)$. *Then there exists an $(n^{O(1)}s(n), d(n) + O(1))$ deterministic circuit family for A.*

The proofs of Ajtai and Ben-Or, and likewise our modifications, use circuits as black boxes, which are combined by NOT-, OR-, and AND-gates. Hence the results above apply to probabilistic $\mathcal{G}$-circuits (defined in the obvious way) as well. We have the following lemma.

Lemma B.8. *Let $\{C\}_{n=1}^{\infty}$ be an $(s(n), d(n))$ probabilistic $\mathcal{G}$-circuit family where D_n has $k(n)$ random variables. Let $\{Y\}_{n=1}^{\infty}$ be such that $Y_n \subseteq \{0,1\}^{k(n)}$. Assume that for all n,* $\mathrm{SEP}(D_n, A_n, \overline{A}_n, \frac{3}{4}, \frac{1}{4}, Y_n)$. *Then there exists an $(n^{O(1)}s(n), d(n) + O(1))$ deterministic $\mathcal{G}$-circuit family for A.*

References

[1] M. Agrawal, V. Arvind, Quasi-linear truth-table reductions to p-selective sets, Theoret. Comput. Sci. 158(1–2) (1996) 361–370 (Note).
[2] M. Ajtai, M. Ben-Or, A theorem on probabilistic constant depth circuits, in: Proceedings of the 16th ACM Symposium on Theory of Computing, 1984, pp. 471–474.
[3] A. Amir, R. Beigel, W. Gasarch, Some connections between bounded query classes and non-uniform complexity, Technical report TR 00-0024, 2000. Available from <www.ecc.uni-trier.de/eccc/>.
[4] A. Amir, W. Gasarch, Polynomial terse sets, Inf. Comput. 77 (April) (1988) 37–56.
[5] J.L. Balcázar, R.V. Book, U. Schöning, The polynomial-time hierarchy and sparse oracles, J. ACM 33 (3) (1986) 603–617.
[6] J.L. Balcázar, R.V. Book, U. Schöning, Sparse sets, lowness and highness, SIAM J. Comput., 15(3) (1986) 739–747 (Prior version appeared in IEEE Symposium on Mathematical Foundations of Computer Sciences, 1984 (MFCS)).
[7] R. Beals, R. Chang, W. Gasarch, J. Torán, On the number of automorphisms of a graph, Chicago J. Theoret. Comput. Sci. 1999. Electronic journal with website <http://www.cs.uchicago.edu/publications/cjtcs/>.
[8] R. Beigel, Query-limited reducibilities, Ph.D. thesis, Stanford University, 1987. Also available as Report No. STAN-CS-88–1221.
[9] R. Beigel, A structural theorem that depends quantitatively on the complexity of SAT, in: Proceedings of the 2nd IEEE Conference Structure in Complexity Theory, IEEE Computer Society Press, 1987, 28–32.
[10] R. Beigel, NP-hard sets are p-superterse unless $R = NP$, Technical report 4, Department of Computer Science, The Johns Hopkins University, 1988.
[11] R. Beigel, Bounded queries to SAT and the Boolean hierarchy, Theoret. Comput. Sci. 84 (1991) 199–223.
[12] R. Beigel, W. Gasarch, J. Gill, J. Owings, Terse, superterse, and verbose sets, Inf. Comput. 103 (1) (1993) 68–85.
[13] R. Beigel, W. Gasarch, M. Kummer, G. Martin, T. McNicholl, F. Stephan, The complexity of ODD_n^A, J. Symbolic Logic (2000) 1–18. (Parts of this appeared in MFCS96 with title On the query complexity of sets).
[14] R. Beigel, M. Kummer, F. Stephan, Approximable sets, Inf. Comput. 120 (2) (1995) 304–314.
[15] R. Beigel, M. Kummer, F. Stephan, Quantifying the amount of verboseness, Inf. Comput. 118(1) (1985) 73–90 (Prior version in Lecture Notes for Computer Science ("Logic at Tver"), vol. 620, pp. 21–32, 1992).
[16] J. Bondy, U. Murty, Graph Theory with Applications, Elsevier, New York, 1977.
[17] R. Book, J. Lutz, D. Martin, The global power of additional queries to random oracles, Inf. Comput. (1995) 49–58.
[18] R.V. Book, K.-I. Ko, On sets truth-table reducible to sparse sets, SIAM J. Comput. 17 (1988) 903–919.
[19] R. Boppana, M. Sipser, The complexity of finite functions, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Algorithms and Complexity, vol. A, MIT Press, Elsevier, The Netherlands, 1990, pp. 757–804.
[20] H. Buhrman, L. Fortnow, Two queries, J. Comput. Syst. Sci. 59 (1999) 182–194.
[21] J. Cai, Lower bounds for constant depth circuits in the presence of help bits. Inf. Process. Lett. 36 (1990) 79–84 (Prior version in IEEE Symposium on Foundations of Computer Sciences, 1989 (FOCS)).
[22] J. Cai, L.A. Hemachandra, Enumerative counting is hard, Inf. Comput. 82 (1989) 34–44 (Prior verison in IEEE Conference on Structure in Complexity Theory, 1988 (STRUCTURES)).
[23] J. Cai, L.A. Hemachandra, A note on enumerative counting, Inf. Process. Lett. 38 (4) (1991) 215–219.
[24] R. Chang, On the structure of bounded queries to arbitrary NP sets, SIAM J. Comput. 21 (4) (1992) 743–754.
[25] R. Chang, J. Kadin, The Boolean hierarchy and the polynomial hierarchy: a closer connection, SIAM J. Comput. 25(2) (1996) 340–354.
[26] R. Chang, J. Kadin, The Boolean hierarchy and the polynomial hierarchy: a closer connection, SIAM J. Comput. 25 (2) (1996) 340–354.
[27] M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman, New York, 1979.
[28] W. Gasarch, The complexity of optimization functions, Technical report 1652, Department of Computer Science, University of Maryland, 1985.
[29] W. Gasarch, A hierarchy of functions with applications to recursive graph theory, Technical report 1651, Department of Computer Science, University of Maryland, 1985.
[30] W. Gasarch, Bounded queries in recursion theory: a survey, in: Proceedings of the 6th IEEE Conference on Structure in Complexity Theory, IEEE Computer Society Press, 1991, pp. 62–78.

[31] W. Gasarch, in: B. Cooper, S. Goncharov (Eds.), Computability and Models (Chapter: Gems in the Field of Bounded Queries), Plenum Press, New York, 2003.
[32] W. Gasarch, M.W. Krentel, K. Rappoport, OptP-completeness as the normal behavior of NP-complete problems, Math. Syst. Theory 28 (1995) 487–514.
[33] W. Gasarch, G. Martin, Progress in Computer Science and Applied Logic, Birkhäuser, Boston, 1999.
[34] J. Goldsmith, D. Joseph, P. Young, Using self-reducibility to characterize polynomial time, Inf. Comput. 104 (1993) 288–308.
[35] J. Håstad, Almost optimal lower bounds for small depth circuits, in: Proceedings of the 18th ACM Symposium on Theory of Computing, 1986, pp. 6–20.
[36] J. Håstad, Computational Limitations of Small-depth Circuits, MIT Press, Cambridge, MA, 1987.
[37] J. Håstad, Almost optimal lower bounds for small depth circuits, in: S. Micali (Ed.), Randomness and Computation, JAI Press, Greenwich, CT, 1989, pp. 143–170.
[38] E. Hemaspaandra, L. Hemaspaandra, H. Hempel, Extending downward collapse from 1-versus-2 to j-versus-$j+1$, in: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 1563, Springer, Berlin, 1999.
[39] L. Hemaspaandra, L. Torenvliet, Optimal advice, Theoret. Comput. Sci. 154 (2003) 367–377.
[40] L. Hemaspaandra, L. Torenvliet, Theory of Semi-feasible Algorithms, EATCS Monographs in Theoretical Computer Science, Springer, Berlin, 2003.
[41] A. Hoene, A. Nickelsen, Counting, selecting, sorting by query-bounded machines, in: Proceedings of the 10th Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 665, Springer, Berlin, 1993.
[42] R. Karp, R. Lipton, Some connections between non-uniform and uniform complexity classes, in: Proceedings of the 12th ACM Symposium on Theory of Computing, 1980, pp. 302–309.
[43] R. Karp, R. Lipton, Turing machines that take advice, Enseign. Math. (1982) 28.
[44] K.-I. Ko, On self-reducibility and weak P-selectivity, J. Comput. Syst. Sci. 26 (1983) 209–221.
[45] M. Krentel, The complexity of optimization problems, J. Comput. Syst. Sci. 36(3) (1988) 490–509 (Prior version in ACM Symposium on Theory of Computation, 1986 (STOC)).
[46] M. Kummer, A proof of Beigel's cardinality conjecture, J. Symbolic Logic 57 (2) (1992) 677–681.
[47] M. Lerman, Degrees of unsolvability, Perspectives in Mathematical Logic, Springer, Berlin, 1983.
[48] W. Merkle, The global power of additional queries to P-random oracles, sicomp 31 (2002) 483–495 (Prior version in ECCC99 and ICALP00).
[49] A. Meyer, M. Paterson, With what frequency are apparently intractable problems difficult? Technical report MIT/LCS/TM-126, MIT Press, 1979.
[50] M. Ogihara, Polynomial-time membership comparable sets, SIAM J. Comput. 24 (1995) 1068–1081.
[51] J.C. Owings Jr., A cardinality version of Beigel's Non-speedup Theorem, J. Symbolic Logic 54 (3) (1989) 761–767.
[52] C.H. Papadimitriou, S.K. Zachos, Two remarks on the complexity of counting, in: Proceedings of the 6th GI Conference on Theoretical Computer Science, Lecture Notes in Computer Science, vol. 145, Springer, Berlin, 1983, pp. 269–276.
[53] N. Pippenger, On simultaneous resource bounds, in: Proceedings of the 20th IEEE Symposium on Foundations of Computer Sciences, 1979, pp.307–311.
[54] H. Rogers Jr., Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.
[55] C.P. Schnorr, Optimal algorithms for self-reducible problems, in: Proceedings of the 3rd ICALP, 1976, pp. 322–337.
[56] U. Schöning, A low and a high hierarchy within NP, J. Comput. Syst. Sci. 27 (1983) 14–28.
[57] U. Schöning, Complexity and Structure, Springer, Berlin, 1985.
[58] U. Schöning, Probabilistic complexity classes and lowness, J. Comput. Syst. Sci. 39 (December) (1989) 84–100.
[59] A. Selman, Analogues of semirecursive sets and effective reducibilities to the study of NP complexity, Inf. Control 52 (1) (1982) 36–51.
[60] D. Sivakumar, On membership comparable sets, J. Comput. Syst. Sci. (1999) 270–280 (Prior version in IEEE Conference on Complexity Theory, 1998 (STRUCTURES)).
[61] R. Smolensky, Algebraic methods in the theory of lower bounds for Boolean circuit complexity, in: Proceedings of the 19th ACM Symposium on Theory of Computing, 1987, pp. 77–82.
[62] R. Soare, Recursively enumerable sets and degrees, Perspectives in Mathematical Logic, Springer, Berlin, 1987.
[63] F. Stephan, personal communication, 1998.

[64] L. Stockmeyer, The polynomial-time hierarchy, Theoret. Comput. Sci. 3 (1976) 1–22.
[65] L. Stockmeyer, A. Meyer, Word problems requiring exponential time, in: Proceedings of the 5th ACM Symposium on Theory of Computing, 1973, pp. 1–9.
[66] T. Tantau, On the power of extra queries to selective languages, Technical report TR 00-77, Electronic Colloquim on Computational Complexity (ECCC), 2000. Available from <www.eccc.uni-trier.de/eccc/>.
[67] S. Toda, PP is as hard as the polynomial-time hierarchy, SIAM J. Comput. (20) (1991) 865–877 (Prior version in IEEE Symposium on Foundations of Computer Sciences, 1989 (FOCS)).
[68] S. Toda, O. Watanabe, Polynomial time 1-Turing reductions from #PH to #P, Theoret. Comput. Sci. 100 (1992) 205–221.
[69] C.B. Wilson, Relativized circuit complexity, J. Comput. Syst. Sci. 31 (2) (1985) 169–181.
[70] C. Wrathall, Complete sets and the polynomial-time hierachy, Theoret. Comput. Sci. 3 (1976) 23–33.
[71] A.C. Yao, Separating the polynomial-time hierarchy by oracles, in: Proceedings of the 26th IEEE Symposium on Foundations of Computer Sciences, 1985, pp. 1–10.
[72] C. Yap, Some consequences of non-uniform conditions on uniform classes, Theoret. Comput. Sci. 26 (1983) 287–300.